НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Інформаційна система для управління Залишками продуктових магазинів»

Виконала: студентка *IV* курсу, групи *ІС-61*

Козлова Ольга Сергіївна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ас. Очеретяний Олександр Костянтинович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *ас. Очеретяний Олександр Костянтинович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *доц. Кисленко Юрій Іванович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Козловій Ользі Сергіївні
(прізвище, ім’я, по батькові)

1. Тема проєкту « *Інформаційна система для управління залишками продуктових магазинів*»
керівник проєкту *Очеретяний Олександр Костянтинович, асистент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до

технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна класів програмного забезпечення

2. Схема структурна послідовності

3. Схема структурна компонентів програмного забезпечення

4. Схема структурна активності

5. Схема структурна варіантів використань

6. Схема бази даних

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Загальні положення	*ас. Очеретяний О.К.*		
Інформаційне забезпечення	*ас. Очеретяний О.К.*		
Математичне забезпечення	*ас. Очеретяний О.К.*		
Програмне та технічне забезпечення	*ас. Очеретяний О.К.*		
Технологічний розділ	*ас. Очеретяний О.К.*		

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*16.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*23.04.2020*	
3.	*Постановка та формалізація задачі*	*23.04.2020*	
4.	*Розробка інформаційного забезпечення*	*30.04.2020*	
5.	*Алгоритмізація задачі*	*05.05.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*10.05.2020*	
7.	*Розробка програмного забезпечення*	*15.05.2020*	
8.	*Налагодження програми*	*01.06.2020*	
9.	*Виконання графічних документів*	*01.06.2020*	
10.	*Оформлення пояснювальної записки*	*01.06.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Ольга Козлова

Керівник Олександр Очеретяний

Сезм №арк	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6113.00.000 ПЗ	Пояснювальна записка	110	
3	А4	ДП 6113.01.000 ТЗ	Технічне завдання	8	
4	А3	ДП 6113.02.000 ССК	Схема структурна класів програмного	7	
			забезпечення		
5	А3	ДП 6113.03.000 ССП	Схема структурна послідовності	1	
6	А3	ДП 6113.04.000 ССК	Схема структурна компонентів	1	
			програмного забезпечення		
7	А3	ДП 6113.05.000 ССА	Схема структурна активності	1	
8	А3	ДП 6113.06.000 ССВ	Схема структурна варіантів використань	1	
9	А3	ДП 6113.07.000 СБД	Схема бази даних	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6113.00.000.00		
Розробн.		Козлова О.С.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Очеретяний О.К.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-61	
Н/контр.		Телишева Т.О.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Інформаційна система для управління залишками продуктових магазинів

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проєкту складається з шести розділів, містить 17 рисунків, 16 таблиць, 2 додатки, 22 джерела.

Дипломний проєкт присвячений створенню інформаційної системи для управління залишками продуктових магазинів.

Ціль розробки: створити програмний продукт, який шляхом програмування на основі правил дозволить індивідуально підбирати для користувачів продукти із магазинів та супермаркетів, які щоденно готуються та мають малий термін придатності, або ж термін придатності яких скоро закінчиться і вони продаються зі знижками. Це продукти, які повинні збуватися магазинами у першу чергу, аби не нести за собою матеріальних збитків для бізнесу та не продукувати нових харчових відходів.

Задачі розробки наступні: проаналізувати існуючі додатки та визначити їх недоліки, розробити додаток для платформи Android, який складається з фронтенд та бекенд частин, є легким для модифікації та масштабування, порівняти два методи реалізації системи, заснованої на правилах, зробити висновки про використання кожного з них, провести функціональне тестування системи.

У розділі інформаційного забезпечення описані вхідні та вихідні дані, представлено структуру бази даних, описано процес інтеграції.

Розділ математичного забезпечення присвячений описанню двох методів розв’язання поставленої задачі: нативного методу із використанням атрибутивної мови та розширених табличних дерев та метод використання двигуна правил Drools, заснованого на алгоритмі Рете.

У розділі програмного забезпечення описано та обґрунтовано вибір засобів програмного забезпечення для програмної реалізації дипломного проєкту. Сформульовано вимоги до програмного забезпечення. Побудовано діаграми класів, діаграму послідовності та діаграми компонентів. Складено таблицю специфікації функцій програмного забезпечення.

У технологічному розділі представлено детальне керівництво користувача з користування створеним додатком, проведено мануальне тестування системи, що представлено у вигляді таблиць тест-кейсів.

РЕКОМЕНДАЦІЙНА СИСТЕМА, ПРАВИЛА, ДОДАТОК, АТРИБУТИВНА МОВА, ДВИГУН ПРАВИЛ, АЛГОРИТМ РЕТЕ

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of six sections, contains 24 pictures, 16 tables, 1 appendix, 22 sources.

The diploma project is concentrated on the creation of an information system for managing the balances of grocery stores.

Development goal: to create a software product that, using programming based on rules, will individually select products for users from stores and supermarkets whose expiration date expires and those, that are on sale. These products should be sold in stores in the first place in order to not to cause material damage to business or produce new food waste.

The development tasks are the following: analyze existing applications and identify their disadvantages, develop an application for the Android platform, which consists of frontend and backend parts, is easy to modify and scale, compare two methods of implementing a rules-based system, sum up using each of them, provide functional testing of the system.

In the section of information support the input and output data is described, database structure is presented and integration process is explained.

In the section of mathematical software, two methods of solving the problem are described: the native method with using of attributive language and extended table trees and the method with using the Drools rule engine based on the Rete algorithm.

The software section describes and substantiates the choice of software for implementation of the diploma project. Software requirements are formulated. Class diagrams, sequence diagram and component diagrams are constructed. Functions specification table is made.

The technological section presents a detailed user guide for using the created application, manual testing of the system, which is presented in the form of tables of test cases.

RECOMMENDATION SYSTEM, ROOLS, APPLICATION, ATTRIBUTIVE LANGUAGE, RULES ENGINE, RETE ALGORITHM

ЗМІСТ

ВСТУП 8

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ 10

1.1 Опис предметного середовища 10

1.1.1 Опис процесу діяльності 11

1.1.2 Опис функціональної моделі 12

1.2 Огляд наявних аналогів 17

1.3 Постановка задачі 19

1.3.1 Призначення розробки 19

1.3.2 Цілі та задачі розробки 19

Висновок до розділу 20

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 21

2.1 Вхідні дані 21

2.2 Вихідні дані 23

2.3 Опис структури бази даних 23

Висновок до розділу 24

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ 25

3.1 Змістовна постановка задачі 25

3.2 Математична постановка задачі 26

3.3 Обґрунтування методу розв’язання 26

3.4 Опис методів розв’язання 27

Висновок до розділу 40

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ 41

4.1 Засоби розробки 41

4.2 Вимоги до технічного забезпечення 43

4.2.1 Загальні вимоги 43

4.3 Архітектура програмного забезпечення 44

4.3.1 Діаграма класів *44*
4.3.2 Діаграма послідовності *44*
4.3.3 Діаграма компонентів *45*
4.3.4 Специфікація функцій *45*
4.4 Опис звітів 50
Висновок до розділу 50
5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ 51
5.1 Керівництво користувача 51
5.2 Випробування програмного продукту 63
5.2.1 Мета випробувань *63*
5.2.2 Загальні положення *63*
5.2.3 Результати випробувань *63*
Висновок до розділу 65
ЗАГАЛЬНІ ВИСНОВКИ 67
ПЕРЕЛІК ПОСИЛАНЬ 69
ДОДАТОК А ..... ERROR! BOOKMARK NOT DEFINED.
ДОДАТОК Б 71

ВСТУП

Дипломний проєкт присвячений створенню інформаційної системи для управління залишками продуктових магазинів. Така система покликана допомагати бізнесу та навколишньому середовищу водночас, адже кількість продуктового сміття, що продукується планетою, збільшується з кожним роком.

На жаль, без перебільшення можна сказати, що харчові відходи – одна з найбільших екологічних проблем, з якою сьогодні стикається людство. Близько 50% всієї їжі, що виробляється у світі щорічно, ніколи не буває вжитою, а натомість викидається на смітники, що несе за собою більше 1 трильйона доларів збитків. Це є масовою ринковою неефективністю, виду якої не існує у інших галузях.

Харчові відходи, перш за все, є неправильними з точки зору моралі, адже наразі близько 800 мільйонів людей лягають спати голодні. Це означає, що кожна дев’ята людина на планеті так чи інакше стикається з нестачею продовольства. Ця проблема не є проблемою локально-віддаленою, як прийнято вважати, адже навіть у королівстві Велика Британія більше 1 мільйона людей звертаються до банку продовольства щорічно, тоді як у США більше 40 мільйонів жителів страждають від нестачі продукції.

Харчові відходи вкрай негативно впливають на навколишнє середовище. Для того, аби виростити той обсяг продукції, який щорічно викидається планетою, необхідна територія, близька за площею до площі Китаю. Це земля, що виснажується, види тварин та рослин, які зникають, корінні популяції, які вимушені залишати рідні території, ґрунти, що деградують – усе це лише для того, аби виростити їжу, яку ми ніколи не з’їмо.

Мало того, що на виробництво цих продуктів йде безліч ресурсів, одним із найцінніших є, наприклад, прісна вода, але коли ця їжа потрапляє на сміттєзвалища і розкладається без доступу кисню, вона продукує метан, який у 23 рази більш смертельний за вуглекислий газ. Якби харчові відходи були

країною, це була б третя за обсягами викиду парникових газів країна після США та Китаю.

Практичне значення одержаних результатів. Проаналізовано методи проєктування та розробки рекомендаційних систем, заснованих на правилах, реалізовано алгоритм проєктування системи з допомогою атрибутивної мови та розширених табличних дерев.

Публікації. Результати роботи були опубліковані у тезах доповідей на науково-технічних конференції кафедри АСОІУ.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

У якості предметної області для інформаційної системи було обрано ринок супермаркетів України, оскільки супермаркети, на відміну від закладів громадського харчування, мають деяку наперед визначену базу продуктів, де кожен продукт має певні атрибути та характеристики, що спрощує використання даної предметної області для побудови деяких узагальнень, абстракцій, правил та рекомендацій.

За даними американської благодійної організації ReFed [1], розподіл харчових відходів за вагою між галузями наступний (Рисунок 1.1):

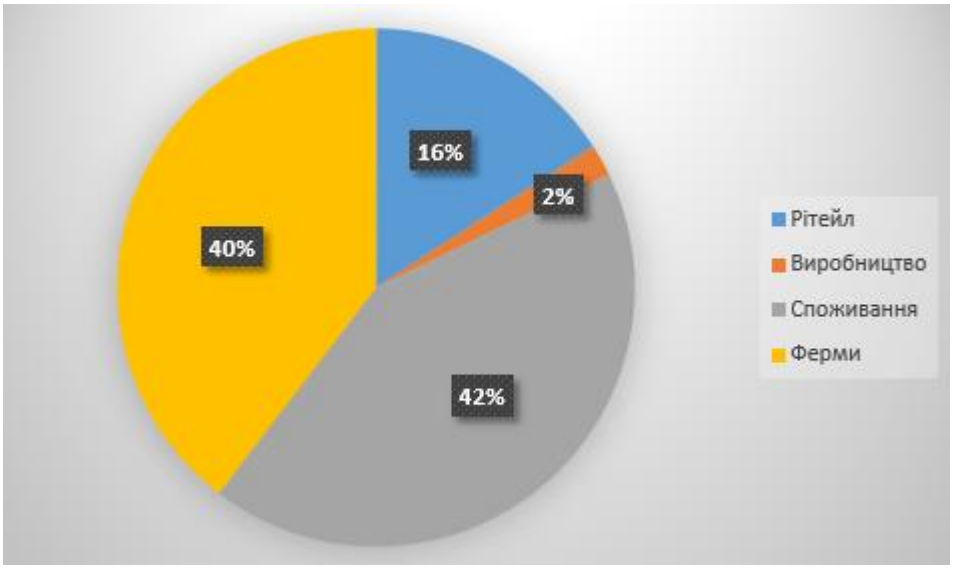


Рисунок 1.1 – Виробники харчових відходів

Найменше сміття продукують фабрики з виробництва продуктів та ферми, де продукти вирощують. Практично навпіл основну масу відходів ділять між собою продавці та споживачі. Практика супермаркетів та закладів публічного харчування відповідає за велику кількість харчових відходів, адже багато продуктів так і не доходять до рук споживача. Продукти можуть бути занадто дорогими для пересічного громадянина, деякі з них вилучаються із ланцюга поставок через їх непрезентабельний вигляд, інколи продукти просто не потрібні користувачеві і він пропускає цілі розділи, не надіючись знайти те, що йому підходить. Також велике значення має грамотність закупок

постачальника, на яку також впливають і зовнішньоекономічні фактори такі як, наприклад, економічна криза, зміна влади, падіння чи зріст валюти, врожай/неврожай сезонних продуктів тощо [2].

Одним із шляхів вирішення проблеми забруднення планети очевидно є використання для цього інформаційних технологій, а саме: розроблення додатків, що допомагають ефективно мінімізувати розміри відходів, створення інформаційних порталів для розповсюдження екологічних ідей, а також для того, щоб допомагати бізнесу і звичайним людям мінімізувати розміри своїх відходів. Якщо ми говоримо, власне, про харчові відходи, то існує безліч способів їх запобігання на кожній фазі [3]:

- Стандартизоване маркування дати для кожного продукту
- Освіта та допомога для користувача
- Оптимізація лінії виробництва

Було обрано другий напрямок із вище наведених. Тобто зменшувати харчові відходи шляхом пропозиції продуктів, що насамперед потрібні користувачеві, або тих, що йому більше пасують, враховуючи звички, дієти, вподобання, та виключенням із пропозицій тих продуктів, які для потенційного користувача є нецікавими з точки зору корисності та практичного застосування.

1.1.1 Опис процесу діяльності

Процес замовлення та закупівлі продуктів є давно автоматизованим, існує безліч рекомендацій щодо організації такої системи. Однак, в цих системах зазвичай немає рекомендаційної складової. Враховуючи цю особливість, була побудована діаграма активності взаємодії користувача із системою, що представлена у графічних матеріалах.

1.1.2 Опис функціональної моделі

Основним актором у системі є користувач. Для нього доступні функції авторизації та автентифікації, у будь-який момент користування додатком він може пройти опитування для того, аби згенерувати або скоригувати ті обмеження, які він накладає на продукти при їх виборі. Ці обмеження будуть використані для генерації правил та створення рекомендацій. Основним варіантом використання для користувача даного додатку є перегляд списку продуктів. Для реалізації цього варіанту користувач може обрати конкретну категорію та підкатегорії для перегляду, ввести назву продукту у пошуковій стрічці або ж переглядати список усіх продуктів відразу. Користувач також має змогу вмикати або вимикати ті обмеження, які він накладає на вибір продуктів. У графічних матеріалах представлено діаграму варіантів використання для користувача.

Відповідно до визначених варіантів використання виявлено функціональні вимоги та встановлена їх пріоритетність, результат наведено у таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги

Актор	Варіант використання	Функціональна вимога	Пріоритет
Користувач	Зареєструватись у додатку	1. Система надає можливість клієнту заповнити реєстраційну форму. 1.1 Система потребує від клієнта заповнити особисту інформацію (логін, пароль,	Важливий Важливий

Продовження таблиці 1.1

Актор	Варіант використання	Функціональна вимога	Пріоритет
		електронна адреса). 1.2 Система вимагає від користувача введення коректного логіну і паролю для прийняття участі у роботі системи.	Важливий
	Увійти в аккаунт	2. Система надає користувачу можливість авторизуватись у системі. 2.1 Система вимагає від користувача введення логіну чи та паролю. 2.2 Система надає можливість перейти на сторінку реєстрації, якщо користувач не зареєстрований.	Важливий
	Переглянути список продуктів	3. Система надає можливість користувачеві переглянути список	Важливий

Продовження таблиці 1.1

Актор	Варіант використання	Функціональна вимога	Пріоритет
		продуктів.	
		3.1 Клієнт має можливість обрати категорію із запропонованих у інтерфейсі.	Важливий
		3.1.1 Після натиснення на назву деякої категорії, система відображає інформацію щодо усіх продуктів, що належать до даної категорії та усіх її підкатегорій.	Важливий
		3.2 У будь-який момент користування розділом категорій чи пошуком за назвою, користувач у власних налаштуваннях може увімкнути або вимкнути обмеження на пошук	Важливий

Продовження таблиці 1.1

Актор	Варіант використання	Функціональна вимога	Пріоритет
		продуктів. 3.2.1 Після зміни режиму перегляду користувачем система завантажує інформацію про продукти заново. 3.3 Клієнт має можливість додати товар у список вподобань. Список вподобань можна переглянути у інтерфейсі користувача.	Важливий Важливий
	Визначити продуктові обмеження	4. Система надає можливість користувачеві визначити індивідуальні продуктові обмеження. 4.1 Після успішної реєстрації користувачеві пропонується	Важливий Важливий

Продовження таблиці 1.1

Актор	Варіант використання	Функціональна вимога	Пріоритет
		анкетна форма із питаннями та варіантами відповідей.	
		4.1.1 Система пропонує користувачеві вказати дієти, яких він дотримується, обираючи їх із запропонованого списку.	Важливий
		4.1.2 Система пропонує користувачеві обрати продукти, яких він уникає, обираючи із запропонованого списку.	Важливий
		4.1.3 Біля деяких варіантів відповідей система розміщує кнопку зі знаком питання, при натисненні на яку	Важливий

1.2 Огляд наявних аналогів

В Україні аналогів подібної програми, або ж програми, яка б дозволяла будь-якими способами боротися із харчовими відходами на території України, поки не існує. Розглянемо світові аналоги додатків для боротьби із продуктовим сміттям:

- ***додаток OLIO*** – популярний переважно на території королівства Велика Британія. Створений виключно для взаємодії сусідів між собою, де люди публікують залишки їжі, аби віддати їх безкоштовно або за символічну ціну, вони стають видимі для користувачів у деякому невеликому радіусі дії і таким чином збуваються поміж сусідів. Додаток [4] простий у використанні і дієвий у боротьбі із продуктовим сміттям, але має обмежений функціонал, не модифікований для використання у регіонах з мовою використання, відмінної від англійської та не взаємодіє з малим та середнім бізнесом;
- ***додаток FoodCloud*** – працює виключно на території Ірландії і є яскравим представником додатку b2b – business to business [5]. Він [6] поєднує між собою виробників продукції різних масштабів та благодійні або соціальні організації, які займаються забезпеченням продукцією незахищених шарів населення. Спеціальний працівник вносить до програми одиниці продукції, які підприємство уже не може продати, повідомлення про появу нових продуктів отримують працівники соціальних організацій. Для звичайного користувача додаток недоступний;
- ***додаток Karma*** – шведський стартап [7], метою якого є надати кінцевому користувачеві доступ до страв зі знижками і місцевим ресторанам та кафе спосіб позбутися страв, що залишилися непроданими впродовж робочого дня. У додатку є можливість оплати

онлайн і система нагород за врятовану їжу, однак працює він виключно на території Швеції

Розроблюваний додаток матиме назву Lavka. Наведемо результати порівняння інформаційної системи із уже існуючими світовими аналогами у вигляді таблиці 1.2.

Таблиця 1.2 – Порівняльна таблиця аналогів

	OLIO	**FOODCLOUD**	**KARMA**	**LAVKA**
Система рекомендацій	-	-	-	+
Покриття українського ринку	-	-	-	+
Взаємодія з супермаркетами	-	+	-	+
Функціонал для поширення продуктів не юридичними особами	+	-	-	-
Взаємодія з благодійними організаціями	-	-	+	-

Як бачимо, усі три додатки допомагають ефективно боротися із продуктовим сміттям, але мають ряд недоліків, один із яких повністю покриває тема даної дипломної роботи – допомога супермаркетам напряму збувати продукти кінцевим користувачам. На території країн СНГ тема продуктового сміття та й сміття взагалі не є передовою, але з кожним роком

популяризується все більше і наявність такого додатку значно спростить задачу розповсюдження екологічних ідей.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням розробки даного додатку є зменшення обсягів продуктового сміття шляхом оптимізації та персоналізації пропозицій на ринку супермаркетів. Якщо продукти, що цікавлять користувача найбільше, з’являться у пошуковій стрічці одними з перших, шанси на те, що вони будуть придбані, значно ростуть. Таким чином, розміри харчових відходів можна значно мінімізувати, адже програма допоможе супермаркетам продати одиниці продукції, які довелося б залишити на звалищі.

1.3.2 Цілі та задачі розробки

Ціль розробки: створити програмний продукт, який шляхом програмування на основі правил [8] дозволить індивідуально підбирати для користувачів продукти із магазинів та супермаркетів, які щоденно готуються та мають малий термін придатності, або ж термін придатності яких скоро закінчиться і вони продаються зі знижками. Це продукти, які повинні збуватися магазинами у першу чергу, аби не нести за собою матеріальних збитків для бізнесу та не продукувати нових харчових відходів.

Задачі розробки:

- проаналізувати існуючі додатки та визначити їх недоліки;
- розробити додаток для платформи Android, який складається з фронтенд та бекенд частин, є легким для модифікації та масштабування;
- порівняти два методи реалізації системи, заснованої на правилах, зробити висновки про використання кожного з них;
- провести функціональне тестування системи

Висновок до розділу

У даному розділі описано предметне середовище інформаційної системи, деталізовано процес діяльності у предметній області. Визначено основні ролі та моделі поведінки, сформульовано функціональні вимоги до програми. Після аналізу конкурентних програм-аналогів було підсумовано їх переваги та недоліки, визначено шляхи подальшого розвитку. Поставлено чітке формулювання задачі, цілі та задачі розробки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Для розробки рекомендаційної системи необхідна достатньо велика кількість тестових даних, аби перевірити систему на ефективність та працездатність. У випадку з предметною областю, що стосується ринку продажу брендових товарів, яка є базовою для створення даної інформаційної системи, для реалістичності вирішено використовувати реальну базу даних продуктів, що продаються у всіх українських мережах супермаркетів.

Такі дані для дипломної роботи були представлені українським макретплейс сервісом listex.info [9] – агрегатором товарів для усіх учасників ланцюга поставок, одним з яких, цілком можливо, може стати розроблюваний додаток. Як результат, написана система інтеграції з даним сервісом, де дані перетворюються у необхідний для використання у рамках додатку вигляд. Усього даних, що стосуються продуктів харчування та напоїв, було отримано більше як 30 тисяч одиниць. Для демонстрації результативності проєктування архітектури системи було також проведено інтеграцію із базою даних продуктів, представленою у вільному доступі міністерством сільського господарства США [10]. У базі даних США розмір даних складає 10 тисяч одиниць продукції.

Для інтеграції із сервісом listex.info використовуються такі основні методи, як отримання ієрархічного дерева категорій продуктів та отримання інформації по усіх продуктах у базі. Ця інформація зберігається у базі listex.info у наступному вигляді (Рисунок 2.1).

Рисунок 2.1 – Діаграма класів продуктового каталогу

Усі товари розділені по категоріях. Кожен товар має список атрибутів, серед яких, зокрема, різноманітні характеристики харчової цінності, вага, розміри, склад продукту, тип та розмір упаковки, коротка та повна назва.

Усі отримані в результаті інтеграції дані приводяться до універсального вигляду.

У результаті інтеграції усі продукти розбиті по ієрархічному дереву категорій, кожен продукт має унікальний код продукту, перші цифри якого відповідають за категоризацію. Наприклад:

55103000 Млинці з фруктами

Перша цифра (5) означає, що продукт знаходиться у категорії «Зернові продукти». Перші дві цифри (55) означають, що продукт знаходиться у підкатегорії «Млинці, вафлі та французькі тости». Перші три цифри (551) означають, продукт знаходиться у підкатегорії «Млинці». Наступні цифри це унікальні цифри продуктового коду. Таким чином, заздалегідь організована категоризація спрощує пошук потрібних продуктів, спрощує генерацію

рекомендації за категоріальною ознакою та передбачає легке інтегрування з новою системою продуктової бази.

2.2 Вихідні дані

Результатом роботи будь-якої рекомендаційної системи є перелік, власне, рекомендацій. Після збору інформації про користувача та виконання усіх прописаних системою правил, користувачеві видається список продуктів, що задовольняє його потребам та вподобанням. У інтерфейсі користувача це список одиниць продукції, розбитий по категоріях або ж знайдений за назвою.

2.3 Опис структури бази даних

Основна сутність бази даних – користувач. Усі користувачі зберігаються у окремій таблиці, що містять інформацію про ідентифікатор, логін, пароль та електронну адресу користувача.

Кожен користувач пов’язаний з продуктами зв’язком багато до багатьох [11], що виражається у існуванні таблиці виду Користувач-Продукт. Допоміжні поля цієї таблиці зберігають інформацію про те, чи додав користувач продукт у список вподобань і чи було обрано даний продукт системою рекомендацій для того, аби показати його користувачеві.

Користувач має пов’язаний з ним список дієт [12], яких він дотримується. Користувач і сутність дієти пов’язані зв’язком багато до багатьох, а інформація, щодо пар користувач-дієта зберігається у окремій таблиці.

Користувач має пов’язаний з ним список інгредієнтів, які він намагається уникати у щоденному вживанні їжі. Сутності користувача та інгредієнта пов’язані зв’язком багато до багатьох, інформація щодо пар користувач-інгредієнт зберігається у окремій таблиці.

Сутність продукту пов’язана із сутностями інгредієнт та харчова цінність зв’язками багато до багатьох.

Таблиці у базі даних та зв’язки між ними відображені у ER (Entity Relationship) діаграмі, що представлена у графічних матеріалах.

Висновок до розділу

У даному розділі описано структуру та процес перетворення вхідних даних, отриманих у результаті інтеграції зі сторонніми сервісами, дається форма категоризації, до якої приводяться усі отримані продуктові категорії. Представлено діаграму відношень сутностей бази даних, кожна сутність описана набором атрибутів та ключів. Описано усі зв’язки між сутностями (один до одного, один до багатьох та багато до багатьох).

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Хоча засновані на правилах системи є усюдисущим інструментом в науці, техніці та повсякденному житті, їх кодування, аналіз і проєктування рідко є предметом більш глибокого теоретичного дослідження; в більшості областей застосування вони просто використовуються (свідомо чи несвідомо) прямим способом, застосовуються для вирішення конкретної проблеми, не звертаючи уваги на такі питання, як їх властивості, мова, оптимізація і т. д.

Системи, засновані на правилах (Rule Based Systems, RBS) [13], являють собою потужний інструмент для специфікації знань в області проєктування і реалізації систем, заснованих на знаннях (Knowledge Based Systems, KBS), в прикладному штучному інтелекті і інженерії знань. Вони також надають універсальну парадигму програмування для таких областей, як моніторинг системи, класифікація ситуацій, діагностика системи, кодування оперативних знань і є ефективним інструментом для реалізації підтримки прийняття рішень.

Незважаючи на широке поширення в робочих системах, теоретичний аналіз систем, заснованих на правилах, мабуть, все ще залишається відкритим питанням для аналізу, методологій проєктування і перевірки теоретичних властивостей. Забезпечення надійності, безпеки, якості та ефективності заснованих на правилах систем вимагає як теоретичного розуміння, так і розробки практичних інструментів.

Хоча основні поняття систем, заснованих на правилах, можна добре пояснити на рівні логіки висловлювань, в разі більш реалістичних, практичних додатків логіка висловлювань стає занадто бідною для кодування знань. Очевидно, що виразна сила мов висловлювань занадто мала для ефективного представлення знань. Отже, оскільки необхідні більш потужні інструменти, стоїть задача ці інструменти сформувати та результативно застосувати їх для

інтерпретації бази знань предметної області у набір правил, які легко реалізувати у програмному вигляді у об’єктно-орієнтованій системі.

3.2 Математична постановка задачі

Нехай існує деяка предметна область, у конкретному випадку маємо набір продуктів харчування із визначеними характеристиками і користувачі, які можуть слідувати дієтам і можуть мати непереносимість окремих інгредієнтів.

Задача: спроєктувати інформаційну систему на основі використання правил, з її допомогою сформулювати такий набір рекомендаційних правил, які оптимально співставляючи продукти та користувачів, у результаті повернуть карту вигляду користувач-список рекомендованих продуктів.

3.3 Обґрунтування методу розв’язання

Для розв’язання задачі проєктування системи на основі правил було вирішено порівняти два різні підходи:

- нативний підхід із використанням атрибутивної мови та розширених табличних дерев [14];
- використання двигуна правил Drools, заснованого на алгоритмі Рете [15]

та порівняти їх швидкість роботи, ефективність, простоту реалізації та впровадження у виробничих системах та бізнес-рішеннях.

3.4 Опис методів розв'язання

Метод атрибутивної мови та розширених табличних дерев

Мови, засновані на концепції атрибутів, мають ряд характеристик, що робить їх практично ідеальними інструментами для практичного уявлення і маніпулювання знаннями; до них належать такі функції:

- введення змінних - атрибути грають роль змінних; один і той же атрибут може приймати різні значення;
- легка специфікація обмежень - оскільки атрибути грають роль змінних, використання різних реляційних символів дозволяє вказувати практично будь-які обмеження;
- параметризація - атрибути можуть також грати роль параметрів, які повинні бути створені в деякій бажаній точці виведення.

У своїй оригінальній книзі Павляк [16] вводить поняття інформаційних систем або систем атрибут-значення, які також визначаються як системи представлення знань (або скорочено KR-системи). Наведемо деякі основні визначення. Нехай U позначає кінцеву непорожню множину (об'єктів), A позначає кінцевий набір атрибутів. Кожен (примітивний) атрибут Ai ∈ A є функцією виду Ai: U → Di, де Di – множина допустимих значень Ai, яка буде називатись доменом Ai.

Таким чином, систему, заснована на знаннях (KR-система), можна подати у вигляді S = (U,A).

У своїй книзі Павляк також дає формальне визначення таблиці рішень. Нехай K = (U, A) - система представлення знань, і нехай C ⊆ A і H ⊆ A - дві підмножини набору всіх атрибутів, які називаються умовними і вирішальними атрибутами. KR-система з виділеними умовними і вирішальними атрибутами називається таблицею рішень виду T = (U,A,C,H).

Розглянемо непорожній, кінцевий набір атрибутів, A = {A1, A2, ..., A n}. Для будь-якого атрибута Ai нехай Di позначає домен цього атрибута, i = 1, 2, ..., n. Домен може бути обмеженим, тобто Di = {d1, d2, ..., dmi} або

нескінченним, наприклад, Di ⊆IR, гдеIR - множина дійсних чисел. Атрибути A1, A2, ..., An позначають деякі властивості, вибрані для вираження знання предметної області аналізованої системи при роботі в певному (локальному) контексті. Далі розглянемо конкретний набір атрибутів H = {H1, H2, ..., Hm} з доменами DH 1, DH 2, ..., DH m, де DH h = {h1, h2, ..., HMH}. Ці атрибути призначені для опису вихідних даних правил, наприклад, висновки, рішення або інші вихідні значення.

Табличну систему можна використовувати як для представлення шаблонів даних, так і для задання правил. У першому випадку відсутні атрибути рішення; конкретні рядки таблиці представляють формули, що описують окремі елементи. У разі правил присутні певні стовпці з атрибутами H рішення (висновку), а інші атрибути грають роль змінних, використовуваних в специфікації попередніх умов.

Нехай маємо набір правил, кожне з яких має форму:

ri:(A1 = di1)∧(A2 = di2)∧...(An = din) → H1 = hi1 ∧H2 = hi2 ∧...Hm = him.

Тепер, беручи до уваги перевагу уніфіковану форму всіх правил в системі, набір правил може бути представлений у прозорій табличній формі, що також визначається як таблиця атрибутивних рішень, представлена формулою (3.1).

$$T = \begin{array}{c|cccccccc} rule & A_1 & A_2 & \dots & A_j & \dots & A_n & H_1 \; H_2 \; \dots \; H_m \\ \hline r_1 & d_{11} & d_{12} & \dots & d_{1j} & \dots & d_{1n} & h_{11} \; h_{12} \; \dots \; h_{1m} \\ r_2 & d_{21} & d_{22} & \dots & d_{2j} & \dots & d_{2n} & h_{21} \; h_{22} \; \dots \; h_{2m} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots \; \vdots \; \vdots \; \vdots \\ r_i & d_{i1} & d_{i2} & \dots & d_{ij} & \dots & d_{in} & h_{i1} \; h_{i2} \; \dots \; h_{im} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots \; \vdots \; \vdots \; \vdots \\ r_k & d_{k1} & d_{k2} & \dots & d_{kj} & \dots & d_{kn} & h_{k1} \; h_{k2} \; \dots \; h_{km} \end{array} \quad (3.1)$$

Вищенаведена таблиця (3.1) представляє k рівномірно структурованих правил прийняття рішень. Використовуючи матричну форму запису, можна

також написати T = [RΦH], де R - самий лівий вектор-стовпець імен правил, Φ - матриця значень умовних атрибутів, а H - найправіша матриця, яка визначає висновки (рішення).

Основне розширення таблиць атрибутивних рішень з атомарними значеннями полягає в тому, щоб допускати неатомарні значення атрибутів, як в умовній частині, так і в заключній частині. В принципі, значенням атрибута може бути будь-яка підмножина його доменів. Будемо вважати, що розглядаються тільки обмежені області.

Нехай u ∈ U позначає об'єкт, Ai ∈ A атрибут і t ⊆ Di деяка підмножина домену атрибуту Ai. Тоді кожне правило матиме наступну форму:

ri: (A1 ∈ ti1)∧(A2 ∈ ti2)∧...(An ∈ tin) → H1 = hi1 ∧H2 = hi2 ∧...Hm = him, а розширена таблиця атрибутивних рішень виглядатиме наступним чином (3.2):

$$
\boldsymbol{T} = \begin{array}{c|cccc|cccc}
rule & A_1 & A_2 & \dots & A_n & H_1 & H_2 & \dots & H_m \\
\hline
r_1 & t_{11} & t_{12} & \dots & t_{1n} & h_{11} & h_{12} & \dots & h_{1m} \\
r_2 & t_{21} & t_{22} & \dots & t_{2n} & h_{21} & h_{22} & \dots & h_{2m} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
r_i & t_{i1} & t_{i2} & \dots & t_{in} & h_{i1} & h_{i2} & \dots & h_{im} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
r_k & t_{k1} & t_{k2} & \dots & t_{kn} & h_{k1} & h_{k2} & \dots & h_{km}
\end{array}
\qquad (3.2)
$$

Із використанням атрибутивної мови для задання системи, заснованої на правилах, самі конкретні правила задаватимуться формулою (3.3).

$$
\begin{aligned}
&rule(i): \psi \wedge \\
&\quad A_1 \in t_1 \wedge A_2 \in t_2 \wedge \dots \wedge A_n \in t_n \\
&\quad \longrightarrow \\
&\quad retract(B_1 = b_1, B_2 = b_2, \dots, B_b = b_b) \\
&\quad assert(C_1 = c_1, C_2 = c_2, \dots, C_c = c_c) \\
&\quad H_1 = h_1, H_2 = h_2, \dots, H_h = h_h \\
&\quad next(j), \quad else(k).
\end{aligned}
\qquad (3.3)
$$

У наведеній вище формулі (3.3):

– (A1 ∈ ti1)∧(A2 ∈ ti2)∧...(An ∈ tin) це формула, що відображає передумову виконання правила;
– B1=b1, …, Bb=bb – факти, що витягаються із бази знань;
– C1=c1, …, Cc=cc – факти, що добавляються у базу знань;
– H1=h1, …, Hh=hh – висновки, кінцеві рішення, що формують вихідні значення правил

Розширені табличні дерева (Extended Table Trees, XTT) об'єднують ідею розширених атрибутивних таблиць з ідеєю дерев рішень [17]. Це забезпечує відносно високу концентрацію специфікацій знань з розширеними таблицями атрибутивних рішень в поєднанні з можливостями управління, включеними в структуру дерев рішень. Фактично, розширені табличні дерева являють собою один з найбільш багатообіцяючих підходів до подання знань, добре відповідаючи вимогам до алгоритмів специфікації, аналізу, перевірки та кодування знань.

Правила в табличній специфікації знань еквівалентні рядкам таблиці. Фактично, кожне правило в формі, заданої специфікацією (3.3), може бути представлено з використанням ряду комірок, що визначають попередні умови і висновок правила. Одне правило представлено у вигляді одного рядка, що має наступну форму (Таблиця 3.1):

Таблиця 3.1 – Розширене табличне дерево

Info		Prec	Retract	Assert	Decision	Ctrl	
I	Ctx	$A_1 \dots A_n$	$B_1 \dots B_b$	$C_1 \dots C_c$	$H_1 \dots H_h$	N	E
i	ψ	$t_{i1} \dots t_{in}$	$b_{i1} \dots b_{ib}$	$c_{i1} \dots c_{ic}$	$h_{i1} \dots h_{ih}$	g_i	e_i

У наведеній вище специфікації:

– I – ідентифікатор правила в таблиці;

– Ctx – контекст правила;

– A1-An – атрибути передумови;

– B1-Bb – атрибути для вилучення;

– C1-Cc – атрибути для вставки;

– H1-Hh – атрибути рішень;

– N – контрольний атрибут, що індикує наступне правило;

– E – контрольний атрибут, що індикує правило для умови «інакше»

Антоні Лігеза [16] у своїй книзі пропонує наступні кроки для того, аби спроєктувати систему, засновану на правилах, з використанням розширених табличних дерев:

– правила – оригінальні правила, представлені у простому словесному викладі;

– атрибути – специфікація атрибутів системи, їх доменів та обмежень;

– діаграма відношень атрибутів;

– логічний дизайн – представлення логіки системи у вигляді розшиненого табличного дерева;

– реалізація

Розглянемо перші 4 кроки проєктування на прикладі обраної предметної області. Нехай маємо користувача, який пройшов опитування та відмітив у ньому, що постійно дотримується кетодієти. Спроєктуємо спрощену систему правил для рекомендації деяких продуктів для даного користувача із усього списку продуктів.

Правила:

- якщо користувач добавив до списку дієт елемент з ідентифікатором 4, він слідує кето дієті;
- якщо дієта – кето дієта, то заборонені категорії – 10, 11, 12;
- якщо дієта – кето дієта, то заборонені ключові слова – «хліб», «цукор», «борошно»;
- якщо продукт не порожній, то виставити категорію продукту, харчову цінність, склад та назву;
- якщо список заборонених категорій не порожній і категорія продукту не порожня, добавити до списку операцій – перевірити заборонені категорії;
- якщо список ключових слів не порожній і назва продукту не порожня і склад продукту не порожній, додати до списку операцій – перевірити ключові слова;
- якщо дієта – кето дієта і харчова цінність продукту не порожня, то додати до списку операцій – перевірити вміст вуглеводів;
- якщо операція – перевірити вміст вуглеводів і вміст вуглеводів не перевищує 5%, додати продукт до списку відображених;
- якщо операція – перевірити заборонені категорії і категорія продукту не міститься у списку заборонених, додати продукт до списку відображених;
- якщо операція – перевірити ключові слова і у назві продукту порожній перетин зі списком ключових слів і у списку складових продуктів порожній перетин зі списком ключових слів, додати продукт до списку відображених;

Атрибути

На етапі складання таблиці атрибутів аналізується предметна область і визначаються основні одиниці, що оперуються правилами. Це їх вхідні та

вихідні дані, а також атрибути, що використовуються всередині правил. Список системних атрибутів предметної області наведений у таблиці 3.2.

Таблиця 3.2 – Список атрибутів

Назва	Позначення	Обмеження
Ідентифікатор дієти	aDI	<1,10>
Продукт	aPR	Product
Дієта	aDIE	Кето дієта, палео дієта, середземноморська…
Харчова цінність	aPNU	Nutrition
Назва продукту	aPN	Літерал
Категорія продукту	aPC	Число
Склад продукту	aPI	Масив інгредієнтів
Заборонені категорії	aRC	Масив чисел
Ключові слова	aRW	Масив літералів
Операція	aOP	Перевірка вуглеводів, перевірка категорій, перевірка ключових слів
Добавляння продукту	aPA	True, false

Діаграма відношень атрибутів

На діаграмі відношень атрибутів (Рисунок 3.1) зображені рівні взаємодії між усіма атрибутами. Зі збільшенням рівня, зменшується рівень абстракції взаємодії. Тобто на нульовому рівні ми зазначаємо, що додавання продукту залежить від самого продукту та ідентифікатора дієти користувача, а на останньому рівні можемо бачити усі атрибути, що мають вагомий вплив.

Рисунок 3.1 – Діаграма відношень атрибутів

Логічний дизайн

Логічний дизайн передбачає побудову так званого дерева рішень, яке показує які правила в якому порядку повинні викликатись, які параметри вони приймають на вхід і який результат видають. Таблиці 3.3-3.7 являють собою детальний опис дерева рішень, де правила поділені на логічні групи, пов’язані переходами.

Таблиця 3.3 – Група 1

Info	Prec	Assert	Ctrl	
I	aDI	aDIE	N	E
1	4	Кето дієта	2.2	1.1

Таблиця 3.4 – Група 2

Info	Prec	Assert		Ctrl	
I	aDIE	aRC	aRW	N	E
2	Кето дієта	10 11 12	Хліб Цукор Борошно	3.3	2.2

Таблиця 3.5 – Група 3

Info	Prec	Assert				Ctrl	
I	aPR	aPC	aPN	aPI	aPNU	N	E
3	Продукт	Характеристики продукту				4.4	3.3

Таблиця 3.6 – Група 4

Info	Prec							Assert	Ctrl	
I	aDIE	aRC	aRW	aPC	aPN	aPI	aPNU	aOP	N	E
4		∅		∅				Перевірка категорій	4.5	4.4
5	Кето дієта						∅	Перевірка вуглеводів	4.6	4.5
6			∅		∅	∅		Перевірка ключових слів	5.7	4.6

Таблиця 3.7 – Група 5

Info	Prec							Decision	Ctrl	
I	aOP	aRC	aRW	aPC	aPN	aPI	aPNU	aPA	N	E
7	Перевірка вуглеводів						Вуглеводи < 5%	true	5.8	5.7
8	Перевірка категорій			⊄aRC				true	5.9	5.8
9	Перевірка ключових слів				∩aRW =∅	∩aRW =∅		true	1.1	5.9

Таким чином, розширене табличне дерево є основою програмної реалізації, воно легко читається та розуміється, керуючись даною специфікацією можна згенерувати ефективні правила для прийняття рішень. Специфікація легко масштабується та модифікується.

Метод двигуна правил Drools

Drools - це механізм правил з відкритим вихідним кодом, написаний на мові Java, який використовує алгоритм Rete [18] для оцінки правил. Реалізація Drools Rete називається ReteOO, що означає, що Drools має поліпшену і оптимізовану реалізацію алгоритму Rete для об'єктно-орієнтованих систем. Drools дозволяє висловлювати правила бізнес-логіки декларативним способом [19]. Ми можемо написати правила, використовуючи не-XML мову, яку досить легко вивчити і зрозуміти. Також існує можливість вставляти код Java безпосередньо в файл правил.

Drools також має наступні переваги:

- активна підтримка спільнотою;
- простота у використанні;
- швидкість виконання;
- популярність серед розробників Java

Drools - це механізм правил, що використовує підхід на основі правил для впровадження експертних систем і його більш правильно класифікувати як система виробничих правил. Система виробничих правил є повною по Тюрінгу з акцентом на подання знань для вираження рекомендаційної логіки і логіки першого порядку в короткій, не двозначній і декларативній манері. Мозок системи - це механізм логічного виводу, здатний масштабуватися до великої кількості правил і фактів. Двигун логічного виводу співставляє факти і дані з правилами, щоб зробити висновки, які призводять до дій. Правило системи - це структура з двох частин, що використовує логіку першого порядку для подання знань: якщо (умова) то (дія).

Правила зберігаються у виробничій пам'яті. Факти розташовуються в робочій області пам'яті, де вони потім вони можуть бути змінені або вилучені. Система з великою кількістю правил і фактів може призвести до того, що багато правил будуть дійсними за твердженням того ж факту, ці правила

знаходяться в конфлікті. Спеціальний механізм управляє порядком виконання цих конфліктуючих правил з використанням стратегії вирішення конфліктів.

Механізм логічного виводу системи правил зберігає свій стан і забезпечує правдивість. Логічні взаємозв’язки можуть бути оголошені діями, що означає, що стан дії залежить від того, чи залишається висновок вірним; коли це більше не вірно, залежна за логікою дія скасовується.

Існує два методи виконання системи правил: пряма і зворотна зв'язність; системи які реалізують обидва способи, називаються гібридними системами. Розуміння цих двох режимів роботи є ключовим щоб зрозуміти, чому системи правил відрізняються від інших систем і як використовувати їх найбільш ефективним способом. Пряме зв'язування є «керованим даними» і, таким чином - факти затверджуються в робочій пам'яті, що призводить до того, що кілька правил паралельно виявляються істинними і приводяться у виконання за допомогою механізму вирішення конфліктів. Drools це двигун з прямим зв'язуванням.

Слово RETE в перекладі з латині означає «мережа». Алгоритм RETE можна розбити на дві частини: компіляція правил і виконання під час роботи програми.

Алгоритм компіляції описує, як створюється ефективна дискримінаційна мережа. У нетехнічних термінах можна сказати, що мережа дискримінації використовується для фільтрації даних. Ідея полягає в тому, щоб фільтрувати дані в міру їх поширення через мережу. У верхній частині мережі вузли будуть мати багато збігів і, просуваючись по мережі нижче, збігів ставатиме менше. У самому низу мережі знаходяться термінальні вузли. Є чотири основних вузла: корінь, 1-вхід, 2-входи і термінальний.

Кореневий вузол - це місце, де всі об'єкти входять в мережу. Звідти вони відразу переходять в ObjectTypeNode. Мета ObjectTypeNode - переконатися, що двигун правил не виконує більше роботи, ніж від нього вимагається. Якщо механізм правил намагатиметься порівняти кожен один вузол із кожним

об'єктом, це буде займати багато циклів для виконання. Щоб зробити процес більш ефективним, двигун повинен передати об'єкт тільки тим вузлам, які відповідають типу об'єкта. Найпростіший спосіб зробити це - створити ObjectTypeNode і зробити так, щоб всі вузли типу 1-вхід і 2-входи спускалися в нього (Рисунок 3.2). Таким чином, якщо додаток отримує новий об'єкт, він не буде поширюватися на вузли для об'єктів іншого типу. У Drools, коли об'єкт затверджується, він отримує список дійсних ObjectTypesNodes через пошук в HashMap з класу об'єкта; якщо цей список не існує, він сканує всі вузли ObjectTypde знаходить дійсні збіги, які кешуються в список. Це дозволяє Drools зіставлятися з будь-яким типом класу, який збігається з екземпляром перевірки.

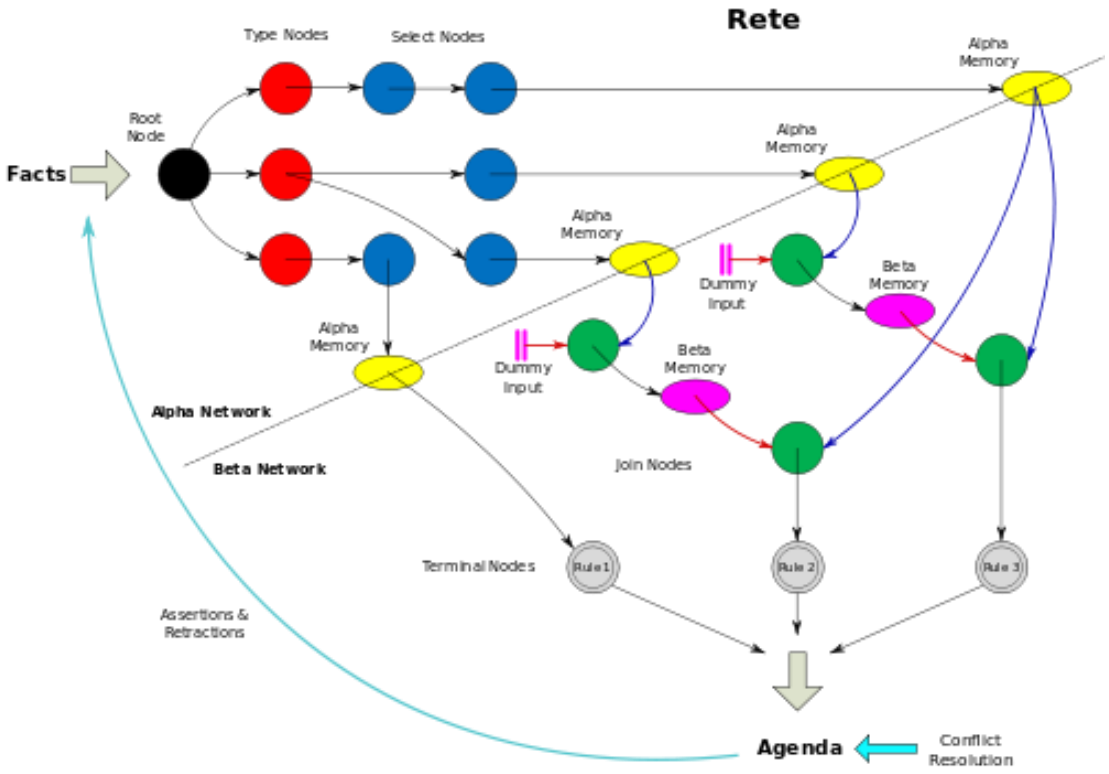


Рисунок 3.2 – Схема роботи алгоритму Рете

Таким чином, двигун правил, використовуючи оптимізований алгоритм Рете, надає також можливість зручної інтеграції у виробничу систему будь-яких масштабів, можливість написання правил декларативною мовою, поєднаною із функціоналом мови Java. Це робить даний інструмент широко розповсюдженим у системах, заснованих на правилах.

Висновок до розділу

У даному розділі здійснено змістовну постановку задачі та сформульовано математичну постановку задачі. Обрано методи розв’язання, вибір кожного з них обґрунтовано. Описано два методи розв’язання поставленої задачі: нативний метод із використанням атрибутивної мови та розширених табличних дерев та метод використання двигуна правил Drools, заснованого на алгоритмі Рете. Для першого методу описано математичний та ідеологічний підхід до проєктування, сформульовано основні терміни атрибутивної мови та розширених табличних дерев, складено атрибутивну таблицю, представлено діаграму відношень атрибутів та побудовано розширене табличне дерево для окремого випадку. Для другого методу описано основні етапи роботи алгоритму Рете.

Підсумовуючи, можна описати переваги та недоліки обох обраних методів (Таблиця 3.8):

Таблиця 3.8 – Переваги та недоліки обраних методів розв’язання

Метод атрибутивних мов та розширених табличних дерев	Метод двигуна правил Drools
+ Високий рівень деталізації + Діаграми та таблиці + Мінімізація помилок + Різні варіанти реалізації - Неможливість оптимізації для ООП - Велика кількість коду	+ Оптимізація для ООП + Стисла декларативна мова + Додаткові функції двигуна - Відсутність перевірки синтаксису - Слабка деталізація

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для реалізації систем, що засновані на правилах, часто застосовують мови, що уже містять в собі логічний апарат (наприклад Prolog [20]), але такі мови зазвичай не підходять для створення великих бізнес-систем, оскільки в них відсутні механізми роботи з мережею, модульність, можливості повторного використання та багато інших. Для написання даної дипломної роботи було обрано мову Java та вирішено розділити фронтенд та бекенд функціонал на два окремі сервери додатків, таким чином реалізуючи архітектуру клієнт-сервер.

Java - мова програмування високого рівня загального призначення [21], зазвичай використовується для створення веб-додатків та мобільних додатків. Дана мова у даному дипломному проєкті використана для написання як бекенд частини так і фронтенд частини у вигляді мобільного Android додатку. Деякі переваги обраної мови програмування [22]:

- простота: Java була розроблена так, щоб її було легше використовувати, компілювати, налагоджувати та вивчати, ніж інші мови програмування. Java набагато простіша за C++, оскільки використовує автоматичне розподілення пам'яті та збирання сміття;
- ООП: Об'єктно-орієнтоване програмування пов'язане з такими поняттями, як клас, об'єкт, успадкування, інкапсуляція, абстракція, поліморфізм тощо, що дозволяє створювати модульні програми та код для багаторазового використання;
- незалежність від платформи: Java програми запускаються на будь-якій платформі та у будь-якому браузері, сумісному з Java. Це дає можливість легко переходити з однієї комп'ютерної системи в іншу;
- безпека: Java - перша мова програмування, яка включає безпеку як невід'ємну частину дизайну. Компілятор, інтерпретатор та

середовище виконання Java розроблялися з урахуванням безпеки. Віртуальна машина Java має унікальний ідентифікатор, який ідентифікує байт-код і перевіряє його перед його запуском

Основою для написання бекенд серверу обрано фреймворк Spring Boot, що по суті є розширенням фреймворку Spring для спрощення налаштувань проєкту і його подальшого розвитку. Spring Boot зводить до мінімуму написання серверних конфігурацій і підтримує вбудований контейнер, який дозволяє веб-додаткам працювати незалежно і без необхідності застосування сторонніх серверів. Центральною частиною Spring є інверсія контролю – абстрактний принцип, набір рекомендацій для написання слабо зв’язаного коду, що означає, що кожен елемент системи повинен бути як можна більше ізольованим від інших, не покладаючись у своїй роботі на детальну реалізацію інших компонентів. Реалізацією цього принципу у Spring є впровадження залежностей.

Для роботи з базою даних використано JPA (Java Persistence API) - це специфікація Java EE і Java SE, що описує систему управління збереженням Java об'єктів в таблиці реляційних баз даних в зручному вигляді. Сама Java не містить реалізації JPA, проте існує багато реалізацій даної специфікації від різних компаній. Однією із найпопулярніших є фреймворк Hibernate, який вирішено використовувати у рамках даного дипломного проєкту. Конфігурація об’єктів та їх зв’язків у якості таблиць бази даних звільняє розробника від написання безлічі однакових SQL запитів для ініціалізації бази даних, звичайних запитів на вилучення та вставку даних тощо, а механізми Hibernate надають широкі можливості для оптимізації запитів, кешування, лінивого завантаження та інших. У якості бази даних використовується відкрита СУБД H2, повністю написана на Java. Незважаючи на малий розмір (дещо більше 1 Мб) , дана СУБД підтримує усі необхідні можливості «з коробки».

Для того, аби доступ до сервера був захищеним від сторонніх запитів, реалізовано механізм автентифікації та авторизації з використанням JWT (JSON Web Token). Це відкритий стандарт для створення токенів доступу, що створюються сервером, підписуються секретним ключем і передаються клієнту, що надалі використовує його для підтвердження своєї особистості. Токен JWT складається з трьох частин: заголовка (header), корисного навантаження (payload) і підпису або даних шифрування. Перші два елементи - це JSON об'єкти певної структури. Третій елемент обчислюється на підставі перших і залежить від обраного алгоритму (HMAC512). Токени можуть бути перекодовані в компактне представлення (JWS / JWE Compact Serialization): до заголовку і корисного навантаження застосовується алгоритм кодування Base64-URL, після чого додається підпис і всі три елементи розділяються крапками («.»).

Клієнтська частина проєкту написана у вигляді мобільного додатку для платформи Android. Android використовується на різних пристроях. Це і смартфони, і планшети, і телевізори, і смарт-годинники і ряд інших гаджетів. За різними підрахунками за 2018 рік цією операційною системою користуються близько 85% власників смартфонів. Для спілкування з сервером використовується бібліотека Retrofit, що є стандартом для мережевої взаємодії у рамках Android додатків. Бібліотека чудово підтримує REST API, легко тестується і налаштовується.

4.2 Вимоги до технічного забезпечення

Загальні вимоги

Оскільки користувач використовує особисто лише клієнтську частину проєкту у вигляді додатку Android, єдиним обмеженням є версія Android. Мінімальна версія – 7.0 Nougat.

4.3 Архітектура програмного забезпечення

Діаграма класів

Бекенд сервер додатку розділений на три незалежні модулі:

- lavka-application – кореневий модуль, у якому зберігаються конфігурації Drools та SpringBoot, налаштовується security складова проєкту, фільтри для перевірки токену тощо;
- lavka-web – модуль, що відповідає за мережевий зв’язок із сервером, тут знаходяться класи-контроллери, що представляють кінцеві url точки для спілкування з сервером через http api;
- lavka-service – модуль, де зберігається уся бізнес логіка сервера, моделі сутностей для бази даних, реалізація основних методів, обраних для вирішення задачі дипломного проєкту

У графічних матеріалах наведені діаграми класів, що логічно розділені на окремі пакети у різних модулях.

Фронтенд частина у вигляді Android-додатку складається із основних елементів, таких як активності та фрагменти, для відображення списку продуктів та категорій реалізовані класи-адаптери. Діаграма класів для Android-додатку представлена у графічних матеріалах.

Діаграма послідовності

Діаграма послідовності детально відображає логіку взаємодії користувача із сервером шляхом використання Android-додатку. У рамках сценарію, користувач напряму взаємодіє із фрагментами та активностями додатку, що у свою чергу відправляють запити до бекенд серверу, що запускає двигун правил, працює з базою даних, виконує сортування та фільтрацію тощо. Діаграма послідовності представлена у графічних матеріалах.

Діаграма компонентів

На діаграмі компонентів, що представлена у графічних матеріалах, зображено деталі взаємодії користувача з додатком на абстрактному рівні модулів. Виділено три основних компоненти: користувач, бекенд сервер та Android-додаток. Кожен з них має кінцеві точки взаємодії та складається з інших компонентів.

Специфікація функцій

У таблиці 4.1 описуються основні функції класів програмного забезпечення.

Таблиця 4.1 – Специфікація функцій

Назва класу	Назва функції	Опис функції
DroolsBeanFactory	private KieFileSystem getKieFileSystem()	Забезпечує контейнер Drools вбудованою у пам’ять файловою системою для автоматичного пошуку файлів правил
	public KieContainer getKieContainer()	Створює Drools контейнер і зв’язує його з усіма доступними ресурсами
	private void getKieRepository()	Встановлює репозиторій у контейнері.
	public KieSession getKieSession()	Створює сесію роботи двигуна правил.

Продовження таблиці 4.1 – 1

Назва класу	Назва функції	Опис функції
JWTAuthenticationFilter	public Authentication attemptAuthentication(HttpServletRequest req, HttpServletResponse res)	Аналізує вхідні дані користувача та передає їх для подальшої обробки менеджеру автентифікації
	protected void successfulAuthentication (HttpServletRequest req, HttpServletResponse res, FilterChain chain, Authentication auth)	Викликається, коли користувач успішно логіниться у систему, генерує JWT токен
JWTAuthorizationFilter	protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain chain)	Фільтрує кожен запит користувача, перевіряючи наявність доступу до деяких ресурсів
	private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request)	Зчитує JWT токен із заголовку запиту користувача і валідує його
UserDetailsServiceImpl	public UserDetails loadUserByUsername(String login)	Завантажує з бази користувача із вказаним логіном, повертає його дані для процесу автентифікації

Продовження таблиці 4.1 – 2

Назва класу	Назва функції	Опис функції
WebSecurityConfiguration	protected void configure(HttpSecurity http)	Налаштовує конфігурацію веб-безпеки сервера, вказує дозволені і заборонені шляхи для запитів, встановлює фільтри автентифікації та авторизації
IntegrationService	default <T> Supplier<T> sendGetHttpRequest(URIBuilder builder, Class<T> clazz)	Відправляє GET запит за вказаним шляхом і вказаним типом відповіді, що очікується
	default <T> List<T> sendListGetHttpRequest(URIBuilder builder, Class<T> clazz)	Відправляє GET запит за вказаним шляхом, коли очікується відповідь у вигляді списку і вказаним типом відповіді, що очікується
	default <T> List<T> sendListPostHttpRequest (URIBuilder builder, Object requestBody, Class<T> clazz)	Відправляє POST запит за вказаним шляхом, коли очікується відповідь у вигляді списку і вказаним типом відповіді, що очікується
	void importProductBase()	Реалізований конкретним класом, метод витягує з інтеграційної бази

Продовження таблиці 4.1 – 3

Назва класу	Назва функції	Опис функції
	void importProductBase()	(серверу) дані про продукти, переформатовує їх у уніфікований вигляд, зручний інформаційній системі та зберігає до бази даних.
	void importCategories()	Реалізований конкретним класом, метод витягує з інтеграційної бази (серверу) ієрархічне дерево категорій, переформатовує їх у уніфікований вигляд, зручний інформаційній системі та зберігає до бази даних.
RestrictionImport	public static DietRestrictionModel getRestrictionsByDietId(int id)	Завантажує заборонені категорії та ключові слова для кожної дієти
Rule<T>	private Consumer<T> fireNextRule()	Викликає наступне правило після того, як дане правило закінчує свою роботу.

Продовження таблиці 4.1 – 4

Назва класу	Назва функції	Опис функції
RuleEngine	public void fireAllRulesForOneUser(User user)	Викликає усі правила (запускає створений ланцюг) для вказаного користувача. Зберігає продукти, що рекомендовані правилами, до бази даних.
AuthController	public User signUp(@RequestBody User user)	Метод реєстрації користувача, шифрує пароль, зберігає користувача до бази даних.
ProductController	public ResponseEntity<List<Category>> getCategories()	Повертає ієрархічне дерево категорій, що зберігається у базі даних
	public ResponseEntity<List<ProductResponse>> getProducts(@PathVariable Long categoryId, @PathVariable Long userId)	Повертає список продуктів, що належать до вказаної категорії та були рекомендовані двигуном правил для конкретного користувача
SurveyController	public <List<QuestionModel>> getQuestions()	Повертає список питань для анкети користувача і варіанти відповідей.

Продовження таблиці 4.1 – 5

Назва класу	Назва функції	Опис функції
	public ResponseEntity<User> sendAnswers(@Request Body SurveyResponse surveyResponse)	Приймає відповіді користувача на задані питання, зберігає інформацію у базі даних, запускає двигун правил для даного користувача, зберігає рекомендовані двигуном продукти та повертає заповнену сутність користувача

4.4 Опис звітів

Результатом роботи двигуна правил по суті є список продуктів, що зв’язком many to many пов’язується з користувачем і надалі є доступний для усіх методів, що повертають сутність користувача у якості відповіді сервера.

Висновок до розділу

У даному розділі описано та обґрунтовано вибір засобів програмного забезпечення для програмної реалізації дипломного проєкту. Сформульовано вимоги до програмного забезпечення. Побудовано діаграми класів, діаграму послідовності та діаграми компонентів. Складено таблицю специфікації функцій програмного забезпечення.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Після запуску додатку користувачем відкривається форма логіну у додаток (Рисунок 5.1). Користувачу необхідно ввести логін та пароль і натиснути кнопку «Увійти».

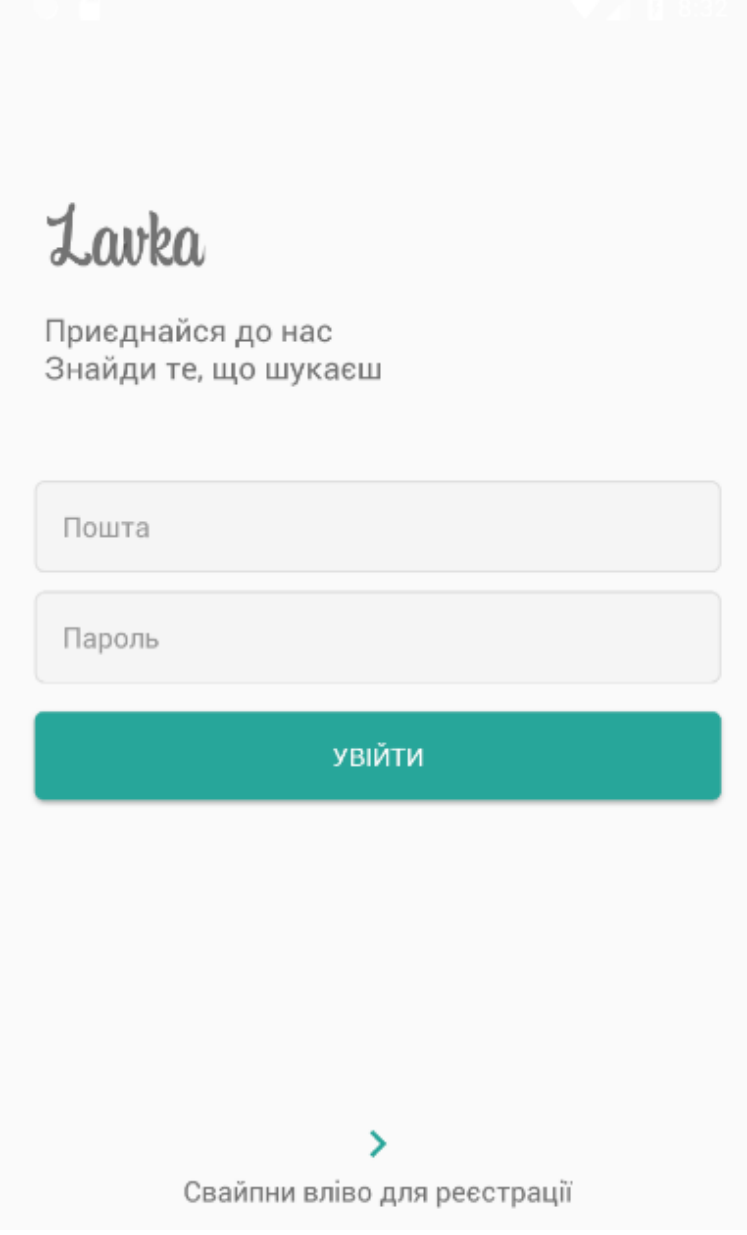


Рисунок 5.1 – Форма логіну у додаток

Якщо користувач не зареєстрований у системі, або вводить невірні логін чи пароль, система показує інформаційне повідомлення у окремому вікні (Рисунок 5.2).

Рисунок 5.2 – Повідомлення про неуспішний логін

Якщо користувач, не зареєстрований у системі, він має можливість відразу перейти на сторінку реєстрації, «свайпнувши» екран вліво (Рисунок 5.3).

Рисунок 5.3 – Форма реєстрації

Після успішної реєстрації, додаток пропонує користувачеві заповнити дані про смакові вподобання, а саме – пройти анкету-опитування (Рисунок 5.4). Користувач має змогу відмітити дієти, яких він дотримується, обрати продукти, яких він уникає у повсякденному вживанні, а також ввести список продуктів через кому (Рисунок 5.5), якщо необхідних йому позицій немає серед запропонованих додатком.

Рисунок 5.4 – Форма опитування користувача

Рисунок 5.5 – Форма для опитування користувача із окремим полем для заповнення індивідуальних продуктів

Після натиснення кнопки «Далі» додаток перенаправляє користувача на головну сторінку перегляду продуктів за категоріями, на якій користувач може бачити наявні у системі категорії продуктів (Рисунок 5.6).

Рисунок 5.6 – Головна сторінка категорій

При натисканні на кожну категорію, з’являється сторінка списку продуктів, обраної категорії, що рекомендовані зареєстрованому користувачеві (Рисунок 5.7).

Рисунок 5.7 – Сторінка перегляду списку продуктів обраної категорії

Користувач може натиснути на будь-який продукт, додаток перенаправляє користувача на сторінку детальної інформації про продукт (Рисунок 5.8). На даній сторінку користувач може переглянути повну назву продукту, його склад, інформацію про харчову цінність, а також може добавити продукт до списку улюблених, натиснувши відповідну кнопку внизу інтерфейсу (Рисунок 5.9).

Рисунок 5.8 – Сторінка інформації про продукт

Рисунок 5.9 – Сторінка інформації про продукт, кнопка для додавання до списку улюблених

У будь-який момент користування додатком, користувач може скористатися боковим навігаційним меню (Рисунок 5.10). У ньому можна перейти на сторінку редагування профілю, перегляду продуктів, ще раз пройти опитування або ж переглянути вподобані продукти.

Рисунок 5.10 – Бокове навігаційне меню

При натисканні на кнопку «Опитування», додаток перенаправляє користувача на форму проходження анкети. При натисканні на кнопку «Профіль», додаток відображає сторінку із інформацією про користувача (Рисунок 5.11), де можна переглянути за змінити логін та пароль, а також увімкнути/вимкнути смакові обмеження на продукти.

Рисунок 5.11 – Сторінка особистого профілю

Для того, аби змінити логін чи пароль, необхідно натиснути на поле логіну чи паролю, з’явиться діалогове вікно (Рисунок 5.12), у якому користувач може виправити значення поля та зберегти нове значення.

Рисунок 5.12 – Форма редагування логіну

Користувач також може переглянути список вподобаних продуктів на окремому фрагменті (Рисунок 5.13).

Рисунок 5.13 – Перегляд вподобань

5.2 Випробування програмного продукту

Мета випробувань

Метою випробувань являється перевірка відповідності функцій інформаційної системи для управління залишками продуктових магазинів вимогам технічного завдання.

Загальні положення

Випробування проводяться на основі наступних документів:

- ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
- ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

Результати випробувань

У процесі мануального тестування був перевірений увесь функціонал інформаційної системи. У наступних таблицях наведений перелік випробувань основних функціональних можливостей (табл. 5.1 – 5.5).

Таблиця **Error! No text of specified style in document.**.1 – Тестування введення логіну та паролю

Мета тесту	Перевірка функції «Увійти в аккаунт»
Початковий стан	Відкрита форма логіну
Вхідні дані	Логін та пароль користувача
Схема проведення тесту	Ввести у поле «Логін» логін існуючого користувача, а у поле «Пароль» – його пароль
Очікуваний результат	Користувач збережений до бази даних
Результуючий стан	Відкрита форма для перегляду продуктів по категоріях, користувач збережений

Таблиця **Error! No text of specified style in document.**.2 – Тестування функції реєстрації

Мета тесту	Перевірка функції «Зареєструватися у додатку»
Початковий стан	Відкрита форма реєстрації
Вхідні дані	Логін, пошта та пароль користувача
Схема проведення тесту	Ввести у поле «Логін» логін нового користувача, у поле «Пошта» електронну пошту, у поле «Пароль» – його пароль, у поле «Повторіть пароль» повторити введений пароль
Очікуваний результат	Користувач збережений у базі даних із введеними параметрами
Результуючий стан	Відкрита форма для проходження опитування у вигляді анкети, користувач існує у базі даних

Таблиця **Error! No text of specified style in document.**.3 – Тестування функції проходження опитування

Мета тесту	Перевірка функції «Визначити продуктові обмеження»
Початковий стан	Відкрита форма опитування
Вхідні дані	Відмічені користувачем чекбокси
Схема проведення тесту	Відмітити чекбокси, натиснути кнопку «Далі»
Очікуваний результат	Вподобання користувача збережені до бази даних, запущені правила, згенеровані та збережені рекомендації
Результуючий стан	Відкрита форма для перегляду продуктів по категоріях, оновлений користувач у базі даних

Таблиця **Error! No text of specified style in document.**.4 – Тестування функції перегляду списку продуктів

Мета тесту	Перевірка функції «Переглянути список продуктів»
Початковий стан	Відкрита форма для перегляду продуктів по категоріях
Вхідні дані	Список категорій
Схема проведення тесту	Натиснути на кнопку будь-якої категорії
Очікуваний результат	Відкрита форма зі списком продуктів обраної категорії
Результуючий стан	Відкрита форма зі списком продуктів обраної категорії

Таблиця **Error! No text of specified style in document.**.5 – Тестування функції додавання продукту до вподобань

Мета тесту	Перевірка функції «Додати продукт до списку улюблених»
Початковий стан	Відкрита форма деталей продукту
Вхідні дані	Інформація про продукт
Схема проведення тесту	Натиснути на кнопку «Додати до улюблених»
Очікуваний результат	У відповідній таблиці багато до багатьох у базі даних новий запис
Результуючий стан	Кнопка змінила свій зовнішній вигляд, користувач у базі даних має оновлений список продуктів вподобань

Висновок до розділу

У даному розділі було представлено детальне керівництво користувача з користування даним додатком, показано приклади екранних форм з поясненнями конкретних дій чи операцій, що можна з ними проводити. Проведено мануальне тестування системи, що представлено у вигляді таблиць тест-кейсів.

ЗАГАЛЬНІ ВИСНОВКИ

У ході виконання дипломного проєкту було розглянуто процес рекомендації продуктів харчування кінцевим користувачам. Були виділені основні етапи, притаманні процесу, та взаємозв'язки між ними, проведений аналіз предметного середовища, описані призначення, мета та задачі, які необхідно вирішити для досягнення поставленої мети.

У розділі інформаційного забезпечення описані вхідні та вихідні дані, процес та мету інтеграції з маркетплейс сервісами публічного та приватного доступу, представлено діаграму відношень сутностей у базі даних та описано усі зв'язки між сутностями.

У розділі математичного забезпечення здійснено змістовну постановку задачі та сформульовано математичну постановку задачі. Обрано методи розв'язання, вибір кожного з них обґрунтовано. Описано два методи розв'язання поставленої задачі: нативний метод із використанням атрибутивної мови та розширених табличних дерев та метод використання двигуна правил Drools, заснованого на алгоритмі Рете. Для першого методу описано математичний та ідеологічний підхід до проєктування, сформульовано основні терміни атрибутивної мови та розширених табличних дерев, складено атрибутивну таблицю, представлено діаграму відношень атрибутів та побудовано розширене табличне дерево для окремого випадку. Для другого методу описано основні етапи роботи алгоритму Рете. Було зроблено висновки про переваги та недоліки використання кожного з методів, що представлені у наглядній таблиці.

У розділі програмного забезпечення описано та обґрунтовано вибір засобів програмного забезпечення для програмної реалізації дипломного проєкту. Сформульовано вимоги до програмного забезпечення. Побудовано діаграми класів для бекенд та фронтенд частин проєкту, діаграму послідовності для повного процесу взаємодії користувача із додатком (починаючи від реєстрації у додатку та закінчуючи переглядом

рекомендованих продуктів) та діаграми компонентів, що детально відображає спланованої архітектуру програмного забезпечення. Складено таблицю специфікації функцій програмного забезпечення, де для кожної функції описано її сигнатуру та деталі використання.

У розділі технологічного забезпечення представлене детальне керівництво користувача та результати функціонального тестування.

ПЕРЕЛІК ПОСИЛАНЬ

1. 27 SOLUTIONS TO FOOD WASTE [Електронний ресурс] // ReFed. – 2020. – Режим доступу до ресурсу: https://www.refed.com/?sort=economic-value-per-ton.
2. Грідін В. Что такое food waste и как от него избавиться? / Володимир Грідін. // Elle. – 2018.
3. Денис А.В., Степанова О.А. ПИЩЕВЫЕ ОТХОДЫ: ПРОБЛЕМЫ И ПУТИ РЕШЕНИЯ // Международный студенческий научный вестник. – 2016. – № 4-5.;
4. Clark T. What is OLIO? [Електронний ресурс] / Tessa Clark // OLIO. – 2019. – Режим доступу до ресурсу: https://olioex.com/about/.
5. Sandhusen R. Marketing / Richard Sandhusen. // Barron's Educational Series. – 2008. – С. 520.
6. Dunphy J. How FoodCloud Works [Електронний ресурс] / John Dunphy // FoodCloud. – 2018. – Режим доступу до ресурсу: https://food.cloud/how-foodcloud-works/.
7. About food waste [Електронний ресурс] // Karma. – 2020. – Режим доступу до ресурсу: https://karma.life/about-food-waste.
8. Crowley J. Knowledge Representation and Reasoning With Rule-Based Production Systems / James Crowley // Intelligent Systems: Reasoning and Recognition / James Crowley., 2018. – (15). – С. 1–20.
9. О проекте [Електронний ресурс] // Listex. – 2018. – Режим доступу до ресурсу: https://listex.info/page/about-listex.
10. USDA. FoodData Central [Електронний ресурс] / USDA – Режим доступу до ресурсу: fdc.nal.usda.gov.
11. Beaulieu A. Learning SQL / Alan Beaulieu. – Sebastopol, United States: O'Reilly Media, Inc, USA, 2009.

12. Van Raes S. 10 Popular Diets Explained [Електронний ресурс] / Sue Van Raes // Chopra Center. – 2018. – Режим доступу до ресурсу: https://chopra.com/articles/10-popular-diets-explained.
13. Wu X. Object-Oriented Modeling of Rule-Based Programming / X. Wu, X. Lin. // Intl Conference on Software Engineering and Knowledge Engineering. – 1997.
14. Antoni L. Logical Foundations for Rule-Based Systems / Ligesa Antoni. – Krakow: AGH University of Science and Technology Press, 2006. – 309 с.
15. Kumar N. Rule based programming with Drools / N. Kumar, D. D Patil, D. M.Wadhai. // (IJCSIT) International Journal of Computer Science and Information Technologies. – 2011. – №2. – С. 1121–1126.
16. Pawlak Z. Theoretical Spects of Reasoning about Data / Pawlak. – London: Kluwer Academic Publishers, 1991.
17. Quinlan, J. R. Induction of Decision Trees. Machine Learning / Quinlan – London: Kluwer Academic Publishers, 1998.
18. Doorenbos R. Production Matching for Large Learning Systems / Robert Doorenbos. // Computer Science Department. – 1995.
19. Оливьери Р. Реализация бизнес-логики при помощи процессора правил Drools [Електронний ресурс] / Рикардо Оливьери // IBM. – 2009. – Режим доступу до ресурсу: https://www.ibm.com/developerworks/ru/library/j-drools/index.html.
20. Маллас Д. Реляционный язык Пролог и его применение. / Джордж Маллас., 1990. – 464 с.
21. Eckel B. Thinking in Java / Bruce Eckel., 2006. – 1482 с.
22. Benefits of Java over Other Programming Languages [Електронний ресурс] // Invensis. – 2015. – Режим доступу до ресурсу: https://www.invensis.net/blog/it/benefits-of-java-over-other-programming-languages

ДОДАТОК А

Тексти програмного коду

Інформаційна система для управління залишками продуктових магазинів

(Найменування програми (документа))

DVD-R

(Вид носія даних)

73 арк,

(Обсяг програми (документа) , арк., Кб)

Київ – 2020 року

```
@Configuration
public class DroolsBeanFactory {

    private KieServices kieServices = KieServices.Factory.get();

    private KieFileSystem getKieFileSystem() {
        KieFileSystem kieFileSystem = kieServices.newKieFileSystem();

kieFileSystem.write(ResourceFactory.newClassPathResource("rules/rules.drl"));
        return kieFileSystem;
    }

    @Bean
    public KieContainer getKieContainer() {
        System.out.println("Container created...");

        getKieRepository();

        KieBuilder kb = kieServices.newKieBuilder(getKieFileSystem());
        kb.buildAll();

        KieModule kieModule = kb.getKieModule();
        return kieServices.newKieContainer(kieModule.getReleaseId());

    }

    private void getKieRepository() {
        final KieRepository kieRepository = kieServices.getRepository();
        kieRepository.addKieModule(kieRepository::getDefaultReleaseId);
    }

    @Bean
    public KieSession getKieSession() {
        System.out.println("session created...");

        return getKieContainer().newKieSession();
    }
}
```

```
public class JWTAuthenticationFilter extends
UsernamePasswordAuthenticationFilter {

    private AuthenticationManager authenticationManager;

    public JWTAuthenticationFilter(AuthenticationManager
authenticationManager) {
        this.authenticationManager = authenticationManager;
    }

    //parse the user's credentials and issue them to the
AuthenticationManager
    @Override
    public Authentication attemptAuthentication(HttpServletRequest req,
                                                HttpServletResponse res)
throws AuthenticationException {
        try {
            User creds = new ObjectMapper()
                    .readValue(req.getInputStream(), User.class);
```

```
            return authenticationManager.authenticate(
                    new UsernamePasswordAuthenticationToken(
                            creds.getLogin(),
                            creds.getPassword(),
                            new ArrayList<>())
            );
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    //method called when a user successfully logs in. We use this method to
generate a JWT for this user
    @Override
    protected void successfulAuthentication(HttpServletRequest req,
                                            HttpServletResponse res,
                                            FilterChain chain,
                                            Authentication auth) throws
IOException {

        String token = JWT.create()

.withSubject(((org.springframework.security.core.userdetails.User)
auth.getPrincipal()).getUsername())
                .withExpiresAt(new Date(System.currentTimeMillis() +
EXPIRATION_TIME))
                .sign(HMAC512(SECRET.getBytes()));
        res.addHeader(HEADER_STRING, TOKEN_PREFIX + token);

        ServletContext servletContext = req.getServletContext();
        WebApplicationContext webApplicationContext =
WebApplicationContextUtils.getWebApplicationContext(servletContext);

        UserRepository userRepository =
webApplicationContext.getBean(UserRepository.class);

        res.getWriter().write(new Gson().toJson(userRepository.findByLogin(
                ((org.springframework.security.core.userdetails.User)
auth.getPrincipal()).getUsername())));
    }
}


public class JWTAuthorizationFilter extends BasicAuthenticationFilter {

    public JWTAuthorizationFilter(AuthenticationManager authManager) {
        super(authManager);
    }

    @Override
    protected void doFilterInternal(HttpServletRequest req,
                                    HttpServletResponse res,
                                    FilterChain chain) throws IOException,
ServletException {
        String header = req.getHeader(HEADER_STRING);

        if (header == null || !header.startsWith(TOKEN_PREFIX)) {
            chain.doFilter(req, res);
            return;
        }
```

```
        UsernamePasswordAuthenticationToken authentication =
getAuthentication(req);

        SecurityContextHolder.getContext().setAuthentication(authentication);
        chain.doFilter(req, res);
    }

    //read the JWT from the Authorization header, and then use JWT to
validate the token
    private UsernamePasswordAuthenticationToken
getAuthentication(HttpServletRequest request) {
        String token = request.getHeader(HEADER_STRING);
        if (token != null) {
            // parse the token
            String user = JWT.require(Algorithm.HMAC512(SECRET.getBytes()))
                    .build()
                    .verify(token.replace(TOKEN_PREFIX, ""))
                    .getSubject();

            if (user != null) {
                return new UsernamePasswordAuthenticationToken(user, null,
new ArrayList<>());
            }
            return null;
        }
        return null;
    }
}



@EnableWebSecurity
public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {

    private UserDetailsServiceImpl userDetailsService;
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    public WebSecurityConfiguration(UserDetailsServiceImpl
userDetailsService, BCryptPasswordEncoder bCryptPasswordEncoder) {
        this.userDetailsService = userDetailsService;
        this.bCryptPasswordEncoder = bCryptPasswordEncoder;
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.cors().and().csrf().disable()
                .authorizeRequests()
                .antMatchers(HttpMethod.POST, SIGN_UP_URL).permitAll()
                .antMatchers(HttpMethod.GET, H2_CONSOLE_URL).permitAll()
                .antMatchers(HttpMethod.POST, H2_CONSOLE_URL).permitAll()
                //.antMatchers(HttpMethod.POST, "/api/**").authenticated()
                .and()
                .addFilter(new
JWTAuthenticationFilter(authenticationManager()))
                .addFilter(new
JWTAuthorizationFilter(authenticationManager()))

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
                .headers().frameOptions().sameOrigin();
    }
```

```
    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception
{

auth.userDetailsService(userDetailsService).passwordEncoder(bCryptPasswordEnc
oder);
    }
}


function boolean checkForbiddenItems(Product product, String
forbiddenIngredients) {
    String[] forbiddenIngredientsList = forbiddenIngredients.split(",");

    for (String ingredient : forbiddenIngredientsList) {

        if
(product.getComposition().toLowerCase().contains(ingredient.toLowerCase())) {
            return false;
        }

        if
(product.getName().toLowerCase().contains(ingredient.toLowerCase())) {
            return false;
        }
    }

    return true;
}

function boolean dietAvailable(List list, int id) {
    return ((List<Object>)list.stream()
    .filter(o -> ((Diet)o).getId() == id)
    .collect(Collectors.toList()))
    .size() > 0;
}

function boolean checkProductCategory(Product product, DietType dietType) {
    for (Long restrictedCategoryId :
dietType.getDietRestrictionModel().getRestrictedCategoriesIds()) {
        if
(String.valueOf(product.getFoodCode()).startsWith(String.valueOf(restrictedCa
tegoryId))) {
            return false;
        }
    }

    return true;
}

function boolean checkProductRestrictedItems(Product product, DietType
dietType) {
    for (String restrictedItem :
dietType.getDietRestrictionModel().getRestrictedItems()) {
        if (product.getName() != null) {
            if
(product.getName().toLowerCase().contains(restrictedItem.toLowerCase())) {
                return false;
            }
        }
```

```
        if (product.getComposition() != null) {
            if (product.getComposition().length() > 0 &&
product.getComposition().toLowerCase().contains(restrictedItem.toLowerCase())
) {
                return false;
            }
        }
    }

    return true;
}

function void addProductToUserList(User user, Product product) {
    user.addProductToList(product);
}

function void deleteProductFromUserList(User user, Product product) {
    user.deleteProductFromList(product);
}

rule "Forbidden products"
salience 1
when
    product : Product (composition != null && name != null)
    user: User (forbiddenIngredients != null, checkForbiddenItems(product,
user.forbiddenIngredients) == false)
then
    deleteProductFromUserList(user, product);
end;

rule "Fat loss add"
salience 9
when
    user: User (dietAvailable(followedDiets, DietType.FAT_LOSS.getId()))
    product : Product (checkProductCategory(product, DietType.FAT_LOSS) ==
true && checkProductRestrictedItems(product, DietType.FAT_LOSS))
then
    addProductToUserList(user, product);
end;

rule "Fat loss delete"
salience 5
when
    user: User (dietAvailable(followedDiets, DietType.FAT_LOSS.getId()))
    product : Product (checkProductCategory(product, DietType.FAT_LOSS) ==
false || checkProductRestrictedItems(product, DietType.FAT_LOSS) == false)
then
    deleteProductFromUserList(user, product);
end;

rule "Vegan add"
salience 10
when
    user: User (dietAvailable(followedDiets, DietType.VEGAN.getId()))
    product : Product (checkProductCategory(product, DietType.VEGAN) == true
&& checkProductRestrictedItems(product, DietType.VEGAN))
then
    addProductToUserList(user, product);
end;

rule "Vegan delete"
salience 5
```

```
when
    user: User (dietAvailable(followedDiets, DietType.VEGAN.getId()))
    product : Product (checkProductCategory(product, DietType.VEGAN) == false
|| checkProductRestrictedItems(product, DietType.VEGAN) == false)
then
    deleteProductFromUserList(user, product);
end;
```

```
@Slf4j
@Service
@PropertySource("keys.properties")
public class ListexIntegrationService implements IntegrationService {

    private static final String SCHEME = "https";
    private static final String LISTEX_HOST = "api.listex.info/v3";
    private static final String GET_CATEGORIES_URL = "/categories";
    private static final String GET_PRODUCT_URL = "/product";

    private static final String API_KEY_PARAM = "apikey";
    private static final String PRODUCT_ID_PARAM = "good_id";

    private static final Long ROOT_CATEGORY_ID = 14001L;

    @Autowired
    private CategoryRepository categoryRepository;

    @Autowired
    private ProductNutritionRepository productNutritionRepository;

    private static Gson gson = new Gson();

    @Value("${listex.key}")
    private String apiKey;

    @Override
    public void importProductBase() {
        List<ListexProduct> productResponses = readProductsFromFile();

        List<RelationProductNutrition> relationProductNutritionList =

parseResponseToRelationProductNutritionList(productResponses);

        productNutritionRepository.saveAll(relationProductNutritionList);
    }

    private List<ListexProduct> readProductsFromFile() {
        List<ListexProduct> productResponses = new ArrayList<>();

        try (Stream<Path> walk =
Files.walk(Paths.get("D:\\diploma\\listex_db"))) {

            List<Path> collect = walk
                    .filter(Files::isRegularFile)
                    .collect(Collectors.toList());

            for (Path filePaths : collect) {
                try (InputStream resource = Files.newInputStream(filePaths);
                     BufferedReader reader = new BufferedReader(
                             new InputStreamReader(resource))) {
```

```
                    String json = reader.lines()
                            .collect(Collectors.joining());

                    Type listexProducts = new
TypeToken<ArrayList<ListexProduct>>() {
                    }.getType();

                    productResponses.addAll(gson.fromJson(json,
listexProducts));
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

        } catch (IOException e) {
            e.printStackTrace();
        }

        return productResponses.stream()
                .filter(distinctByKey(ListexProduct::getProductId))
                .collect(Collectors.toList());
    }

    private <T> Predicate<T> distinctByKey(Function<? super T, ?>
keyExtractor) {
        Set<Object> seen = ConcurrentHashMap.newKeySet();
        return t -> seen.add(keyExtractor.apply(t));
    }

    private List<RelationProductNutrition>
parseResponseToRelationProductNutritionList(List<ListexProduct>
productResponses) {
        List<Category> categories = (List<Category>)
categoryRepository.findAll();

        List<RelationProductNutrition> relationProductNutritionList = new
ArrayList<>();

        for (ListexProduct listexProduct : productResponses) {
            List<ListexProductAttribute> attributes =
listexProduct.getAttributes();

            Product product = new Product(listexProduct.getProductId());

            Map<String, String> productNutritions = new HashMap<>();

            for (ListexProductAttribute attribute : attributes) {
                switch (attribute.getAttributeId().toString()) {
                    case "71":

product.setExpirationMonths(attribute.getAttributeValue());
                        break;
                    case "3969":

product.setExpirationDays(attribute.getAttributeValue());
                        break;
                    case "2481":
                        product.setShortName(attribute.getAttributeValue());
                        break;
                    case "2479":
                        product.setName(attribute.getAttributeValue());
                        break;
```

```
                    case "2484":
product.setComposition(attribute.getAttributeValue());
                        break;
                    case "5":
productNutritions.put(NutritionFact.NutritionFacts.PROTEIN.getCode(),
attribute.getAttributeValue());
                        break;
                    case "1":
productNutritions.put(NutritionFact.NutritionFacts.FAT.getCode(),
attribute.getAttributeValue());
                        break;
                    case "6":
productNutritions.put(NutritionFact.NutritionFacts.CARBS.getCode(),
attribute.getAttributeValue());
                        break;
                    case "34":
                        if
(attribute.getAttributeValueType().equals("ккал/100г")) {
productNutritions.put(NutritionFact.NutritionFacts.ENERGY.getCode(),
attribute.getAttributeValue());
                        }
                        break;
                }
            }

            ListexCategory listexProductCategory =
listexProduct.getCategories().get(0);
            Category category = categories.stream()
                    .filter(c ->
c.getCategoryNumber().equals(listexProductCategory.getCategoryId()))
                    .findAny().orElse(new Category());

            product.setCategoryId(category.getCategoryNumber());

product.setFoodCode(Long.valueOf(category.getCategoryId().toString() +
product.getId().toString()));
            product.setImagePath(listexProduct.getImagePath());

            for (NutritionFact nutritionFact :
NutritionFact.NutritionFacts.getNutritionFacts()) {
                String nutritionValue =
productNutritions.get(nutritionFact.getId().toString());

                if (nutritionValue != null) {
                    relationProductNutritionList.add(new
RelationProductNutrition(product, nutritionFact, nutritionValue));
                }
            }
        }

        return relationProductNutritionList;
    }

    @Override
    public void importCategories() {
        URIBuilder builder = new URIBuilder();
```

```
builder.setScheme(SCHEME).setHost(LISTEX_HOST).setPath(GET_CATEGORIES_URL)
                .setParameter(API_KEY_PARAM, apiKey);

        ListexCategoriesResponse categoriesResponse =
sendGetHttpRequest(builder, ListexCategoriesResponse.class).get();

        List<ListexCategory> categoryList = categoriesResponse.getResult();

        List<Category> thirdLevelCategories = new ArrayList<>();

        long idCounter = 10;
        for (ListexCategory category : categoryList) {
            if (category.getParentCategoryId().equals(ROOT_CATEGORY_ID)) {
                thirdLevelCategories.add(new
Category(category.getCategoryId(),
                        ++idCounter,
                        category.getCategoryName(),
                        null,
                        new ArrayList<>()));
            }
        }

        idCounter = 10;
        for (Category category : thirdLevelCategories) {
            for (ListexCategory listexCategory : categoryList) {
                if
(listexCategory.getParentCategoryId().equals(category.getCategoryNumber())) {
                    category.getSubCategories().add(new
Category(listexCategory.getCategoryId(),
                            Long.valueOf(category.getCategoryId().toString()
+ ++idCounter),
                            listexCategory.getCategoryName(),
                            category,
                            new ArrayList<>()));
                }
            }

            idCounter = 10;
        }

        categoryRepository.saveAll(thirdLevelCategories);
    }
}


@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@SequenceGenerator(name = "user_seq", allocationSize = 1)
@JsonInclude(JsonInclude.Include.NON_NULL)
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"user_seq")
    private Long id;

    private String login;

    private String password;
```

```
    @LazyCollection(LazyCollectionOption.FALSE)
    @ManyToMany
    @JoinTable(
            name = "user_diet",
            joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "diet_id"))
    private List<Diet> followedDiets;

    private boolean supportsSeasonality;

    @LazyCollection(LazyCollectionOption.FALSE)
    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    @Where(clause = "shown = true")
    @JsonManagedReference
    private List<RelationUserProduct> userProducts;

    @LazyCollection(LazyCollectionOption.FALSE)
    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    @Where(clause = "liked = true")
    @JsonManagedReference
    private List<RelationUserProduct> likedProducts;

    @Lob
    private String forbiddenIngredients;

    @Transient
    @JsonIgnore
    private List<Product> ruleProducts;

    public synchronized void addProductToList(Product product) {
        if (ruleProducts == null) {
            ruleProducts = new ArrayList<>();
        }

        if (!ruleProducts.contains(product)) {
            ruleProducts.add(product);
        }
    }

    public synchronized void deleteProductFromList(Product product) {
        if (ruleProducts != null) {
            ruleProducts.remove(product);
        }
    }

    //TODO: add likes

    @Transient
    private Map<Attribute, Object> attributeObjectMap = new HashMap<>();
}



@Entity
@SequenceGenerator(name = "user_prod_seq", allocationSize = 1)
public class RelationUserProduct {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"user_prod_seq")
    private Long id;
```

```
    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product product;

    @ManyToOne
    @JoinColumn(name = "user_id")
    @JsonBackReference
    private User user;

    private boolean shown;

    private boolean liked;

    public RelationUserProduct(User user, Product product) {
        this.user = user;
        this.product = product;
        this.shown = true;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Product getProduct() {
        return product;
    }

    public void setProduct(Product product) {
        this.product = product;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public boolean isShown() {
        return shown;
    }

    public void setShown(boolean shown) {
        this.shown = shown;
    }

    public RelationUserProduct(Product product, User user, boolean shown) {
        this.product = product;
        this.user = user;
        this.shown = shown;
    }

    public RelationUserProduct() {
    }

    public boolean isLiked() {
```

```
        return liked;
    }

    public void setLiked(boolean liked) {
        this.liked = liked;
    }
}


public class Rule<T> {
    private int groupId;
    private int ruleId;

    private Consumer<T> predicate;
    private Consumer<T> assertion;

    private int nextRuleGroupId;
    private int nextRuleId;

    private Consumer<T> fire = input -> predicate.andThen(assertion)
            .andThen(fireNextRule())
            .accept(input);

    private Consumer<T> fireNextRule() {
        if (nextRuleGroupId != 0 && nextRuleId != 0) {
            return RuleEngine.rules.stream()
                    .filter(r -> r.getGroupId() == nextRuleGroupId)
                    .filter(r -> r.getRuleId() == nextRuleId)
                    .findFirst()
                    .orElse(new Rule())
                    .getFire();
        } else {
            return t -> {
            };
        }
    }

    public Rule(int groupId, int ruleId, Consumer<T> predicate, Consumer<T>
assertion, int nextRuleGroupId, int nextRuleId) {
        this.groupId = groupId;
        this.ruleId = ruleId;
        this.predicate = predicate;
        this.assertion = assertion;
        this.nextRuleGroupId = nextRuleGroupId;
        this.nextRuleId = nextRuleId;
    }

    public Rule() {
    }

    public int getGroupId() {
        return groupId;
    }

    public int getRuleId() {
        return ruleId;
    }


    public Consumer<T> getFire() {
        return fire;
    }
```

```
}


public enum Operation {
    CHECK_CARBS(userProductPair -> {
        System.out.println("carbs");

        Diet.DietType diet = (Diet.DietType)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aDIE);

        if (diet.equals(Diet.DietType.KETO)) {

        }
    }),

    CHECK_CATEGORIES(userProductPair -> {
        System.out.println("check");

        List<Long> restrictedCategoriesId = (List<Long>)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aRC);

        Long categoryId = userProductPair.getProduct().getFoodCode();

        boolean isAllowed = true;

        for (Long restrictedCategoryId : restrictedCategoriesId) {
            if
(String.valueOf(categoryId).startsWith(String.valueOf(restrictedCategoryId)))
{
                isAllowed = false;
            }
        }

        if (isAllowed) {

userProductPair.getUser().addProductToList(userProductPair.getProduct());
        }

    }),

    CHECK_WORDS(userProductPair -> {
        System.out.println("check words");

        List<String> restrictedWords = (List<String>)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aRW);

        boolean isAllowed = true;

        for (String restrictedItem : restrictedWords) {
            if
(userProductPair.getProduct().getName().toLowerCase().contains(restrictedItem
.toLowerCase())) {
                isAllowed = false;
            }

//            for (Ingredient ingredient :
userProductPair.getProduct().getComposition()) {
//                if
(ingredient.getName().toLowerCase().contains(restrictedItem.toLowerCase())) {
//                    isAllowed = false;
//                }
```

```
//             }
        }

        if (isAllowed) {

userProductPair.getUser().addProductToList(userProductPair.getProduct());
        }
    });

    private Consumer<UserProductPair> consumerFunction;

    Operation(Consumer<UserProductPair> consumerFunction) {
        this.consumerFunction = consumerFunction;
    }

    public Consumer<UserProductPair> getConsumerFunction() {
        return consumerFunction;
    }
}



public class KetoDietRules {
    private static Consumer<UserProductPair> ketoDietIdIndicatePredicate =
userProductPair -> {
        if (userProductPair.getUser().getFollowedDiets()
                .stream()
                .filter(d -> d.getId().equals(4L))
                .findAny()
                .orElse(null) != null) {

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aDI, 4);
        }
    };

    private static Consumer<UserProductPair> ketoDietIndicateAssertion =
userProductPair -> userProductPair.getUser()
            .getAttributeObjectMap().put(Attribute.aDIE, Diet.DietType.KETO);

    private static Rule<UserProductPair> ketoDietIndicateRule = new Rule(1,
1,
            ketoDietIdIndicatePredicate, ketoDietIndicateAssertion, 2, 2);

    //-----

    private static Consumer<UserProductPair> ketoDietRestrictionPredicate =
userProductPair -> {

    };

    private static Consumer<UserProductPair> ketoDietRestrictionAssertion =
userProductPair -> {
        if (userProductPair.getUser()

.getAttributeObjectMap().get(Attribute.aDIE).equals(Diet.DietType.KETO)) {


userProductPair.getUser().getAttributeObjectMap().put(Attribute.aRC,

RestrictionImport.getRestrictionsByDietId(Diet.DietType.KETO.getId()).getRest
rictedCategoriesIds());
```

```
userProductPair.getUser().getAttributeObjectMap().put(Attribute.aRW,

RestrictionImport.getRestrictionsByDietId(Diet.DietType.KETO.getId()).getRest
rictedItems());
        }
    };

    private static Rule<UserProductPair> ketoDietRestrictionRule = new
Rule(2, 2,
            ketoDietRestrictionPredicate, ketoDietRestrictionAssertion, 3,
3);

    public static List<Rule> getRules() {
        return List.of(ketoDietIndicateRule, ketoDietRestrictionRule);
    }
}


public class GeneralRules {
    private static Consumer<UserProductPair> productCharsPredicate =
userProductPair -> {
        if (userProductPair.getProduct() != null) {
            userProductPair.getUser()
                    .getAttributeObjectMap().put(Attribute.aPR,
userProductPair.getProduct());
        }
    };

    private static Consumer<UserProductPair> productCharsAssertion =
userProductPair -> {
        Product product = (Product)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aPR);

        if (product != null) {

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aPC,
product.getFoodCode());

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aPN,
product.getName());

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aPI,
product.getComposition());

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aPNU,
product.getNutritionFacts());
        }
    };

    private static Rule<UserProductPair> productCharsRule = new Rule(3, 3,
            productCharsPredicate, productCharsAssertion, 4, 4);

    //------

    private static Consumer<UserProductPair> carbsOperationPredicate =
userProductPair -> {

    };

    private static Consumer<UserProductPair> carbsOperationAssertion =
```

```
userProductPair -> {
        Object productNutrition =
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aPNU);
        Object diet =
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aDIE);

        if (diet.equals(Diet.DietType.KETO) && productNutrition != null) {
            List<Operation> operationList = (List<Operation>)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aOP);

            if (operationList == null) {
                operationList = new ArrayList<>();
                operationList.add(Operation.CHECK_CARBS);

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aOP,
operationList);
            } else {
                if (!operationList.contains(Operation.CHECK_CARBS)) {
                    operationList.add(Operation.CHECK_CARBS);
                }
            }
        }
    };

    private static Rule<UserProductPair> carbsOperationsRule = new Rule(4, 4,
            carbsOperationPredicate, carbsOperationAssertion, 4, 5);

    //------

    private static Consumer<UserProductPair> categoriesOperationPredicate =
userProductPair -> {

    };

    private static Consumer<UserProductPair> categoriesOperationAssertion =
userProductPair -> {
        Object restrictedCategories =
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aRC);
        Object productCategories =
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aPC);

        if (restrictedCategories != null && productCategories != null) {
            List<Operation> operationList = (List<Operation>)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aOP);

            if (operationList == null) {
                operationList = new ArrayList<>();
                operationList.add(Operation.CHECK_CATEGORIES);

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aOP,
operationList);
            } else {
                if (!operationList.contains(Operation.CHECK_CATEGORIES)) {
                    operationList.add(Operation.CHECK_CATEGORIES);
                }
            }
        }
    };

    private static Rule<UserProductPair> categoriesOperationRule = new
Rule(4, 5,
            categoriesOperationPredicate, categoriesOperationAssertion, 4,
```

```
6);


    //------

    private static Consumer<UserProductPair> wordsOperationPredicate =
userProductPair -> {

    };

    private static Consumer<UserProductPair> wordsOperationAssertion =
userProductPair -> {
        Object restrictedWords =
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aRW);
        Object productName =
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aPN);
        Object productIngredients =
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aPI);

        if (restrictedWords != null && productName != null &&
productIngredients != null) {
            List<Operation> operationList = (List<Operation>)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aOP);

            if (operationList == null) {
                operationList = new ArrayList<>();
                operationList.add(Operation.CHECK_WORDS);

userProductPair.getUser().getAttributeObjectMap().put(Attribute.aOP,
operationList);
            } else {
                if (!operationList.contains(Operation.CHECK_WORDS)) {
                    operationList.add(Operation.CHECK_WORDS);
                }
            }
        }
    };

    private static Rule<UserProductPair> wordsOperationsRule = new Rule(4, 6,
            wordsOperationPredicate, wordsOperationAssertion, 5, 7);

    //------

    private static Consumer<UserProductPair> operationPredicate =
userProductPair -> {

    };

    private static Consumer<UserProductPair> operationAssertion =
userProductPair -> {
        List<Operation> operationList = (List<Operation>)
userProductPair.getUser().getAttributeObjectMap().get(Attribute.aOP);

        for (Operation operation : operationList) {
            operation.getConsumerFunction().accept(userProductPair);
        }
    };

    private static Rule<UserProductPair> operationsRule = new Rule(5, 7,
            operationPredicate, operationAssertion, 0, 0);
```

```
    public static List<Rule> getRules() {
        return List.of(productCharsRule, carbsOperationsRule,
categoriesOperationRule, wordsOperationsRule, operationsRule);
    }
}
```

```
public class RuleEngine {

    @Autowired
    private ProductRepository productRepository;

    static List<Rule> rules;

    static {
        rules = new ArrayList<>();

        rules.addAll(KetoDietRules.getRules());

        rules.addAll(GeneralRules.getRules());
    }


    public void fireAllRulesForOneUser(User user) {
        List<Product> productList = (List<Product>)
productRepository.findAll();

        for (Product product : productList) {
            UserProductPair userProductPair = new UserProductPair(user,
product);

            rules.get(0).getFire().accept(userProductPair);
        }
    }
}
```

```
@Slf4j
@RestController
@RequestMapping(path = "/api/auth")
public class AuthController {

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @Autowired
    private UserRepository userRepository;

    @PostMapping("/sign-up")
    public User signUp(@RequestBody User user) {
        user.setPassword(bCryptPasswordEncoder.encode
                (user.getPassword()));

        userRepository.save(user);

        log.info("AuthController.signUp | User with login {} is signed in",
user.getLogin());
```

```
            return user;
        }
    }



    @RestController
    @RequestMapping("/api/food")
    public class ProductController {

        @Qualifier("listexIntegrationService")
        @Autowired
        private IntegrationService integrationService;

        @Autowired
        private RuleEngine ruleEngine;

        @Autowired
        private UserRepository userRepository;

        @Autowired
        private CategoryRepository categoryRepository;

        @GetMapping("/test/{userId}")
        public User test(@PathVariable Long userId) {
            User user = userRepository.findById(userId).orElse(new User());

            ruleEngine.fireAllRulesForOneUser(user);

            return user;
        }

        @GetMapping(value = "/categories")
        public ResponseEntity<List<Category>> getCategories() {
            //integrationService.importCategories();

            //integrationService.importProductBase();

            List<Category> all = (List<Category>) categoryRepository.findAll();
            all.add(new Category(0L));

            return ResponseEntity.ok(all);
        }

        @GetMapping(value = "/{categoryId}/products")
        public ResponseEntity<List<ProductResponse>> getProducts(@PathVariable
    Long categoryId) {
            User user = userRepository.findById(1L).orElse(new User());

            List<ProductResponse> products = user.getUserProducts().stream()
                    .map(RelationUserProduct::getProduct)
                    .filter(p -> p.getCategoryId().equals(categoryId))
                    .distinct()
                    .map(ProductResponse::new)
                    .collect(Collectors.toList());

            return ResponseEntity.ok(products);
        }

    }
```

```
@Slf4j
@RestController
@RequestMapping("/api/survey")
public class SurveyController {

    @Autowired
    private DietRepository dietRepository;

    @Autowired
    private IngredientRepository ingredientRepository;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private KieSession session;

    @Autowired
    private ProductRepository productRepository;

    @GetMapping("/questions")
    public ResponseEntity<List<QuestionModel>> getQuestions() {
        QuestionModel diets = new
QuestionModel(Question.DIETS_QUESTION.getText(),
                ((List<Diet>) dietRepository.findAll())
                        .stream()
                        .map(Diet::getName)
                        .collect(Collectors.toList()));

        QuestionModel forbiddenProducts = new
QuestionModel(Question.FORBIDDEN_PRODUCTS_QUESTION.getText(),
                ingredientRepository.findAllByRareIsTrue()
                        .stream()
                        .map(Ingredient::getName)
                        .collect(Collectors.toList()));

        return new ResponseEntity<>(List.of(diets, forbiddenProducts),
HttpStatus.OK);
    }

    @PostMapping("/sendAnswers")
    public UserResponse sendAnswers(@RequestBody SurveyResponse
surveyResponse) {
        log.info("SurveyController.sendAnswers | Rules engine is started for
userId {}", surveyResponse.getUserId());

        Iterable<Product> all = productRepository.findAll();

        for (Product product : all) {
            session.insert(product);
        }

        User user =
userRepository.findById(surveyResponse.getUserId()).orElse(null);

        if (user != null) {
            FactHandle factHandle = session.getFactHandle(user);
            if (factHandle != null) {
                session.delete(factHandle);
            }
```

```
            surveyResponse.getQuestionAnswers().forEach((k, v) -> {
                if (k.equals(Question.DIETS_QUESTION.getText())) {

                    List<Diet> userDiets = new ArrayList<>();

                    for (String dietName : v) {
                        Diet.DietType.getDiets()
                                .stream()
                                .filter(d -> d.getName().equals(dietName))
                                .findFirst()
                                .ifPresent(userDiets::add);
                    }

                    user.getFollowedDiets().clear();
                    user.getFollowedDiets().addAll(userDiets);
                } else {
                    user.setForbiddenIngredients(String.join(",", v));
                }
            });

            session.insert(user);
            session.fireAllRules();

            List<Product> collect =
user.getRuleProducts().stream().distinct().collect(Collectors.toList());
            for (Product product : collect) {
                RelationUserProduct relationUserProduct = new
RelationUserProduct(product, user, true);

                if (user.getUserProducts() == null) {
                    user.setUserProducts(new ArrayList<>());
                }

                user.getUserProducts().add(relationUserProduct);
            }

            User savedUser = userRepository.save(user);

            log.info("SurveyController.sendAnswers | Rules engine is finished
for userId {}", surveyResponse.getUserId());

            return new UserResponse(savedUser);
        } else {

            log.error("SurveyController.sendAnswers | Rules engine failed for
userId {}", surveyResponse.getUserId());

            return new UserResponse();
        }
    }
}
```

```
@Slf4j
@RestController
@RequestMapping("/api/user")
public class UserController {

    @Autowired
```

```
    private UserRepository userRepository;

    @GetMapping("/{login}")
    public UserResponse getUserByLogin(@PathVariable String login) {
        User user = userRepository.findByLogin(login);

        log.info("UserController.getUserByLogin | User with login {} is
logged in", login);

        return new UserResponse(user);
    }

    @PostMapping("/save")
    public UserResponse updateUser(@RequestBody User user) {
        userRepository.save(user);

        log.info("UserController.updateUser | User with login {} is updated",
user.getLogin());

        return new UserResponse(user);
    }
}



public interface LavkaClient {

    @GET("api/food/categories")
    Call<List<Category>> categories();

    @GET("api/food/{categoryId}/products")
    Call<List<Product>> products(@Path("categoryId") Long categoryId);

    @POST("api/user/save")
    Call<User> updateUser(@Body User user);

    @POST("api/auth/sign-up")
    Call<User> signUp(@Body User user);

    @POST("api/survey/sendAnswers")
    Call<User> submitSurvey(@Body SurveyResponse surveyResponse);

    @GET("api/user/{login}")
    Call<User> login(@Path("login") String login);

}


public class RestService {
    private static final String API_BASE_URL = "http://10.0.2.2:8080/";

    public static final LavkaClient client;

    static {
        OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

        Gson gson = new GsonBuilder()
                .setLenient()
                .excludeFieldsWithoutExposeAnnotation()
                .create();
```

```
        Retrofit.Builder builder = new Retrofit.Builder()
                .baseUrl(API_BASE_URL)
                .addConverterFactory(GsonConverterFactory.create(gson));

        Retrofit retrofit = builder
                .client(httpClient.build())
                .build();

        client = retrofit.create(LavkaClient.class);
    }
}


public class Singleton {

    private User user;

    private List<Category> categories;

    private Singleton() {
    }

    private static class SingletonHolder {
        private static final Singleton instance = new Singleton();
    }

    public static Singleton getInstance() {
        return SingletonHolder.instance;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public List<Product> getUserProductsByCategoryId(Long categoryId) {
        return user.getProducts().stream()
                .filter(p ->
String.valueOf(p.getCategoryId()).startsWith(String.valueOf(categoryId)))
                .collect(Collectors.toList());
    }

    public List<Product> getUserProducts() {
        return user.getProducts();
    }

    public User getUser() {
        return user;
    }

    public List<Category> getCategories() {
        return categories;
    }

    public void setCategories(List<Category> categories) {
        this.categories = categories;
    }
}


public class CategoriesFragment extends Fragment implements
AdapterView.OnItemSelectedListener {
```

```
    GridView gridView;

    public CategoriesFragment() {

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_product, container,
false);
        gridView = v.findViewById(R.id.gridView);

        getCategoriesList();

        return v;
    }

    private void getCategoriesList() {
        if (Singleton.getInstance().getCategories() == null) {
            Call<List<Category>> call = RestService.client.categories();

            call.enqueue(new Callback<List<Category>>() {

                @Override
                public void onResponse(Call<List<Category>> call,
Response<List<Category>> response) {
                    List<Category> body = response.body();

                    loopCategories(body, null);

                    List<Category> categories = body.stream()
                            .filter(c -> c.getSubCategories().size() > 0)
                            .collect(Collectors.toList());

                    Singleton.getInstance().setCategories(categories);

                    setupCategoriesAdapter(categories);
                }

                @Override
                public void onFailure(Call<List<Category>> call, Throwable t)
{
                    System.out.println(t.getMessage());
                }
            });

        } else {
            setupCategoriesAdapter(Singleton.getInstance().getCategories());
        }
    }

    private static void loopCategories(List<Category> categories, Category
parentCategory) {
        for (Category category : categories) {
            category.setParentCategory(parentCategory);

            if (category.getSubCategories() != null) {
                loopCategories(category.getSubCategories(), category);
            }
        }
    }
```

```
    private void setupCategoriesAdapter(List<Category> categories) {
        gridView.setAdapter(new CategoryAdapter(getActivity(), categories));
    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {

    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
}



public class CategoryAdapter extends BaseAdapter {
    private FragmentActivity context;
    private List<Category> categoryList;

    public CategoryAdapter(Context context, List<Category> products) {
        this.context = (FragmentActivity) context;
        this.categoryList = products;
    }

    @Override
    public int getCount() {
        return categoryList.size();
    }

    @Override
    public Object getItem(int position) {
        return categoryList.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Button button;

        if (convertView == null) {
            button = new Button(context);
            button.setText(categoryList.get(position).getCategoryName());
        } else {
            button = (Button) convertView;
        }
        button.setId(position);

        button.setOnClickListener(v -> {
            Long categoryId = categoryList.get(position).getCategoryId();

            if (categoryId == 0L) {
                ListProductListFragment listProductListFragment = new
ListProductListFragment();
```

```
context.getSupportFragmentManager().beginTransaction().replace(R.id.fragment_
container,
                        listProductListFragment).commit();
            } else {
                ProductListFragment productListFragment = new
ProductListFragment();

                Bundle args = new Bundle();
                args.putLong("categoryId", categoryId);
                productListFragment.setArguments(args);


context.getSupportFragmentManager().beginTransaction().replace(R.id.fragment_
container,
                        productListFragment).commit();
            }

        });

        return button;
    }
}


public class LikedFragment extends Fragment {

    GridView gridView;

    public LikedFragment() {

    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_liked, container, false);
        gridView = v.findViewById(R.id.gridView);

        getProductsList();

        return v;
    }

    private void getProductsList() {
        Call<List<Product>> call = RestService.client.products(14002L);

        call.enqueue(new Callback<List<Product>>() {

            @Override
            public void onResponse(Call<List<Product>> call,
Response<List<Product>> response) {
                List<Product> body = response.body();

                setupProductsAdapter(Collections.singletonList(body.get(0)));
            }

            @Override
            public void onFailure(Call<List<Product>> call, Throwable t) {
                System.out.println(t.getMessage());
```

```
            }
        });

    }

    private void setupProductsAdapter(List<Product> products) {
        gridView.setAdapter(new ProductAdapter(getActivity(), products));
    }
}


public class ListProductListFragment extends Fragment {

    public ListProductListFragment() {

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_list_product_list,
container, false);

        List<String> userProducts = Singleton.getInstance().getUserProducts()
                .stream()
                .map(Product::getShortName)
                .filter(Objects::nonNull)
                .collect(Collectors.toList());

        ListView productsList = v.findViewById(R.id.productsList);

        ArrayAdapter<String> adapter = new ArrayAdapter(v.getContext(),
                android.R.layout.simple_list_item_1, userProducts);

        productsList.setAdapter(adapter);

        productsList.setOnItemClickListener((parent, v1, position, id) -> {

        });

        return v;
    }

}


public class LoginFragment extends Fragment {


    public LoginFragment() {

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_login, container, false);

        return v;
    }

}
```

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager viewPager = findViewById(R.id.viewPager);

        AuthenticationPagerAdapter pagerAdapter = new
AuthenticationPagerAdapter(getSupportFragmentManager());
        pagerAdapter.addFragmet(new LoginFragment());
        pagerAdapter.addFragmet(new RegisterFragment());
        viewPager.setAdapter(pagerAdapter);
    }

    public void login(View view) {
        EditText login = findViewById(R.id.login_login);

        loginUserAndChangeActivity(login.getText().toString());
    }

    public void goToSurvey(View view) {
        EditText login = findViewById(R.id.register_login);
        EditText password = findViewById(R.id.register_password);

        User user = new User(login.getText().toString(),
password.getText().toString());

        signUpUserAndChangeActivity(user);
    }

    private void signUpUserAndChangeActivity(User user) {
        Call<User> call = RestService.client.signUp(user);

        call.enqueue(new Callback<User>() {

            @Override
            public void onResponse(Call<User> call, Response<User> response)
{
                User user = response.body();

                user.setRestrictionsOn(true);
                Singleton.getInstance().setUser(user);
            }

            @Override
            public void onFailure(Call<User> call, Throwable t) {

            }
        });

        Intent intent = new Intent(this, SurveyActivity.class);
        startActivity(intent);
    }

    private void onLoginFail() {
        AlertDialog alertDialog = new AlertDialog.Builder(this).create();

        alertDialog.setTitle("Помилка");
```

```
            alertDialog.setMessage("Логін чи пароль введені неправильно!");


            alertDialog.setButton(DialogInterface.BUTTON_POSITIVE, "OK", (dialog,
    which) -> {

            });

            alertDialog.show();
        }

        private void loginUserAndChangeActivity(String login) {
            Call<User> call = RestService.client.login(login);

            call.enqueue(new Callback<User>() {

                @Override
                public void onResponse(Call<User> call, Response<User> response)
    {
                    if (response.isSuccessful()) {
                        User user = response.body();

                        user.setRestrictionsOn(true);
                        Singleton.getInstance().setUser(user);

                        //openNavigation();
                    } else {
                        onLoginFail();
                    }
                }

                @Override
                public void onFailure(Call<User> call, Throwable t) {
                    onLoginFail();
                }
            });

            Intent intent = new Intent(this, NavigationActivity.class);
            startActivity(intent);
        }

        class AuthenticationPagerAdapter extends FragmentPagerAdapter {
            private ArrayList<Fragment> fragmentList = new ArrayList<>();

            public AuthenticationPagerAdapter(FragmentManager fm) {
                super(fm);
            }

            @Override
            public Fragment getItem(int i) {
                return fragmentList.get(i);
            }

            @Override
            public int getCount() {
                return fragmentList.size();
            }

            void addFragmet(Fragment fragment) {
                fragmentList.add(fragment);
            }
```

```
    }
}


public class NavigationActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {
    private DrawerLayout drawer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_navigation);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
drawer, toolbar,
                R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        if (savedInstanceState == null) {

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_containe
r,
                    new CategoriesFragment()).commit();
            navigationView.setCheckedItem(R.id.nav_products);
        }
    }


    @Override
    public void onBackPressed() {
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.navigation, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();

        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
```

```
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.nav_profile:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_containe
r,
                        new ProfileFragment()).commit();
                break;
            case R.id.nav_products:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_containe
r,
                        new CategoriesFragment()).commit();
                break;
            case R.id.nav_survey:
                Intent intent = new Intent(this, SurveyActivity.class);

                startActivity(intent);

                break;
            case R.id.nav_liked:

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_containe
r,
                        new LikedFragment()).commit();
                break;
        }

        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
}


public class ProductAdapter extends BaseAdapter {
    private FragmentActivity context;
    private List<Product> productList;

    public ProductAdapter(Context context, List<Product> products) {
        this.context = (FragmentActivity) context;
        this.productList = products;
    }

    @Override
    public int getCount() {
        return productList.size();
    }

    @Override
    public Object getItem(int position) {
        return productList.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }
```

```
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View gridView;

        if (convertView == null) {

            gridView = inflater.inflate(R.layout.product_grid_item, null);

            TextView textView = gridView
                    .findViewById(R.id.product_name);

            String name;
            if (productList.get(position).getShortName() != null) {
                name = productList.get(position).getShortName();
            } else if (productList.get(position).getName() != null) {
                name = productList.get(position).getName();
            } else {
                name = productList.get(position).getComposition();
            }

            textView.setText(name);

            ImageView imageView = gridView
                    .findViewById(R.id.product_img);

            imageView.setOnClickListener(v -> {
                ProductDetailsFragment productDetailsFragment = new
ProductDetailsFragment();

                Bundle args = new Bundle();
                args.putSerializable("product", productList.get(position));
                productDetailsFragment.setArguments(args);


context.getSupportFragmentManager().beginTransaction().replace(R.id.fragment_
container,
                        productDetailsFragment).commit();
            });

            String imagePath = productList.get(position).getImagePath();
            new DownloadImageTask(imageView).execute(imagePath);
        } else {
            gridView = convertView;
        }

        return gridView;
    }

    @SuppressLint("StaticFieldLeak")
    public static class DownloadImageTask extends AsyncTask<String, Void,
Bitmap> {
        ImageView bmImage;

        DownloadImageTask(ImageView bmImage) {
            this.bmImage = bmImage;
        }

        protected Bitmap doInBackground(String... urls) {
            String urldisplay = urls[0];
```

```
            Bitmap bmp = null;

            try {
                InputStream in = new java.net.URL(urldisplay).openStream();
                bmp = BitmapFactory.decodeStream(in);
            } catch (Exception e) {
                e.printStackTrace();
            }

            return bmp;
        }

        protected void onPostExecute(Bitmap result) {
            bmImage.setImageBitmap(result);
        }
    }
}


public class ProductDetailsFragment extends Fragment {


    public ProductDetailsFragment() {

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        Product product = (Product)
getArguments().getSerializable("product");

        View v = inflater.inflate(R.layout.fragment_product_details,
container, false);

        ImageView productImageView = v.findViewById(R.id.product_img);
        new
ProductAdapter.DownloadImageTask(productImageView).execute(product.getImagePa
th());

        TextView productNameView = v.findViewById(R.id.product_name);
        productNameView.setText(product.getShortName());

        TextView fullProductNameView = v.findViewById(R.id.full_name);
        fullProductNameView.setText(product.getName());

        TextView expirationView = v.findViewById(R.id.expiration);
        expirationView.setText(product.getExpiration());

        TextView compositionView = v.findViewById(R.id.composition);
        compositionView.setText(product.getComposition());

        TextView proteinsView = v.findViewById(R.id.proteins);
        proteinsView.setText(product.getNutritionFacts().get("Protein"));

        TextView carbsView = v.findViewById(R.id.carbs);
        carbsView.setText(product.getNutritionFacts().get("Carbs"));

        TextView fatsView = v.findViewById(R.id.fats);
        fatsView.setText(product.getNutritionFacts().get("Fat"));

        TextView energyView = v.findViewById(R.id.energy);
```

```
            energyView.setText(product.getNutritionFacts().get("Energy"));

            return v;
        }

    }


    public class ProductListFragment extends Fragment {

        GridView gridView;

        public ProductListFragment() {

        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
            View v = inflater.inflate(R.layout.fragment_product_list, container,
    false);
            gridView = v.findViewById(R.id.gridView);

            long categoryId = getArguments().getLong("categoryId");

            List<Product> userProductsByCategoryId =

    Singleton.getInstance().getUserProductsByCategoryId(categoryId);

            if (userProductsByCategoryId.isEmpty()) {
                TextView textView = v.findViewById(R.id.noProductsText);
                textView.setVisibility(View.VISIBLE);
            } else {
                setupProductsAdapter(userProductsByCategoryId);
            }

            return v;
        }

        private void setupProductsAdapter(List<Product> products) {
            gridView.setAdapter(new ProductAdapter(getActivity(), products));
        }

    }


    public class ProfileFragment extends Fragment {

        AlertDialog alertDialog;
        EditText editText;

        public ProfileFragment() {
        }


        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {

            View v = inflater.inflate(R.layout.fragment_profile, container,
    false);
```

```
        User user = Singleton.getInstance().getUser();

        CheckBox restrictionsCheckBox = v.findViewById(R.id.restrictions);
        restrictionsCheckBox.setChecked(user.isRestrictionsOn());

        restrictionsCheckBox.setOnCheckedChangeListener(
                (buttonView, isChecked) -> {
                    user.setRestrictionsOn(isChecked);

                    Singleton.getInstance().setUser(user);
                }
        );

        TextView loginView = v.findViewById(R.id.login);
        loginView.setText(user.getLogin());

        loginView.setOnClickListener(v1 -> {
            alertDialog = new
AlertDialog.Builder(this.getContext()).create();
            editText = new EditText(this.getContext());

            editText.setText(loginView.getText());

            alertDialog.setTitle("Редагувати логін");
            alertDialog.setView(editText);

            alertDialog.setButton(DialogInterface.BUTTON_POSITIVE,
"Зберегти", (dialog, which) -> {
                loginView.setText(editText.getText());

                user.setLogin(editText.getText().toString());

                updateUser(user);
            });

            alertDialog.show();
        });

        return v;
    }

    private void updateUser(User user) {
        Call<User> call = RestService.client.updateUser(user);

        call.enqueue(new Callback<User>() {

            @Override
            public void onResponse(Call<User> call, Response<User> response)
{
                Singleton.getInstance().setUser(response.body());
            }

            @Override
            public void onFailure(Call<User> call, Throwable t) {

            }
        });
    }
}
```

```
public class RegisterFragment extends Fragment {


    public RegisterFragment() {

    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_register, container,
false);
    }

}


public class SurveyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_survey);

        User user = Singleton.getInstance().getUser();
        if (user != null) {
            List<String> diets = user.getDiets();
            List<String> restrictedItems = user.getRestrictedItems();

            List<CheckBox> dietCheckboxes = this.getDietCheckboxes();
            for (CheckBox checkBox : dietCheckboxes) {
                if (diets.contains(checkBox.getText().toString())) {
                    checkBox.setChecked(true);
                }
            }

            List<CheckBox> productCheckboxes = this.getProductCheckboxes();
            for (CheckBox checkBox : productCheckboxes) {
                if (restrictedItems.contains(checkBox.getText().toString()))
{
                    checkBox.setChecked(true);
                }
            }
        }
    }

    public void onSurveySubmit(View view) {
        Map<String, List<String>> surveyAnswers = fillAnswers();

        User user = Singleton.getInstance().getUser();

        sendSurveyAnswers(user, surveyAnswers);
    }

    private void sendSurveyAnswers(User user, Map<String, List<String>>
surveyAnswers) {
        SurveyResponse surveyResponse = new SurveyResponse(user.getId(),
surveyAnswers);

        Call<User> call = RestService.client.submitSurvey(surveyResponse);
```

```
        call.enqueue(new Callback<User>() {

            @Override
            public void onResponse(Call<User> call, Response<User> response)
{
                User user = response.body();

                user.setRestrictionsOn(true);
                Singleton.getInstance().setUser(user);
            }

            @Override
            public void onFailure(Call<User> call, Throwable t) {
                System.out.println("Error");
            }
        });

        Intent intent = new Intent(this, NavigationActivity.class);
        startActivity(intent);
    }

    private Map<String, List<String>> fillAnswers() {
        Map<String, List<String>> surveyAnswers = new HashMap<>();

        List<String> dietsAnswers = new ArrayList<>();

        List<CheckBox> dietsCheckboxes = this.getDietCheckboxes();

        for (CheckBox checkBox : dietsCheckboxes) {
            if (checkBox.isChecked()) {
                dietsAnswers.add(checkBox.getText().toString());
            }
        }

        surveyAnswers.put("Оберіть пункти, які найкраще вас описують:",
dietsAnswers);

        List<String> productsAnswers = new ArrayList<>();

        List<CheckBox> productsCheckboxes = this.getProductCheckboxes();

        for (CheckBox checkBox : productsCheckboxes) {
            if (checkBox.isChecked()) {
                productsAnswers.add(checkBox.getText().toString());
            }
        }

        surveyAnswers.put("Оберіть продукти, яких ви намагаєтесь уникати:",
productsAnswers);

        return surveyAnswers;
    }

    private List<CheckBox> getDietCheckboxes() {
        List<CheckBox> dietsCheckboxes = new ArrayList<>();

        dietsCheckboxes.add(findViewById(R.id.weightloss));
        dietsCheckboxes.add(findViewById(R.id.paleo));
        dietsCheckboxes.add(findViewById(R.id.mediterrian));
        dietsCheckboxes.add(findViewById(R.id.keto));
        dietsCheckboxes.add(findViewById(R.id.diabet));
        dietsCheckboxes.add(findViewById(R.id.vegan));
```

```
            dietsCheckboxes.add(findViewById(R.id.vegetatian));
            dietsCheckboxes.add(findViewById(R.id.pesk));
            dietsCheckboxes.add(findViewById(R.id.glutenfree));
            dietsCheckboxes.add(findViewById(R.id.milkfree));
            dietsCheckboxes.add(findViewById(R.id.saltfree));

            return dietsCheckboxes;
        }

        private List<CheckBox> getProductCheckboxes() {
            List<CheckBox> productsCheckboxes = new ArrayList<>();

            productsCheckboxes.add(findViewById(R.id.eggplant));
            productsCheckboxes.add(findViewById(R.id.mayonaise));
            productsCheckboxes.add(findViewById(R.id.mushrooms));
            productsCheckboxes.add(findViewById(R.id.nuts));
            productsCheckboxes.add(findViewById(R.id.onion));
            productsCheckboxes.add(findViewById(R.id.seafood));
            productsCheckboxes.add(findViewById(R.id.soy));
            productsCheckboxes.add(findViewById(R.id.olive));
            productsCheckboxes.add(findViewById(R.id.eggs));
            productsCheckboxes.add(findViewById(R.id.subproducts));
            productsCheckboxes.add(findViewById(R.id.citrus));

            return productsCheckboxes;
        }
    }
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО
Керівник проєкту
_____ *Олександр Очеретяний*
(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)
“14” квітня 2020 р.

Інформаційна система для управління залишками продуктових магазинів

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6113.01.000 ТЗ*

на 8 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ…….. 3

1.1 Повне найменування системи……..…….. 3

1.2 Найменування організації-замовника та організацій-учасників робіт…… 3

1.3 Перелік документів, на підставі яких створюється система……... 3

1.4 Планові терміни початку і закінчення роботи зі створення системи……... 3

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ…….. 4

2.1 Призначення системи…….. 4

2.2 Цілі створення системи…….. 4

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ…… 5

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ……..... 6

4.1 Вимоги до функціональних характеристик…….. 6

4.2 Вимоги до надійності…….. 6

4.3 Вимоги до складу і параметрів технічних засобів…… 6

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ……. 7

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ……. 8

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи

Інформаційна система для управління залишками продуктових магазинів.

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовник системи: ТОВ «Українські інформаційні технології».

Розробник системи: студентка 4 курсу ФІОТ, кафедри АСОІУ Козлова Ольга Сергіївна.

1.3 Перелік документів на підставі яких створюється система

Підставою для розробки системи є завдання на переддипломну практику.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку робіт над системою – 18.05.2020.

Плановий термін завершення робіт над системою – 01.06.2020.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Призначенням розробки даного додатку є зменшення обсягів продуктового сміття шляхом оптимізації та персоналізації пропозицій на ринку супермаркетів.

2.2 Цілі створення системи

Ціль розробки: створити програмний продукт, який шляхом програмування на основі правил дозволить індивідуально підбирати для користувачів продукти із магазинів та супермаркетів, які щоденно готуються та мають малий термін придатності, або ж термін придатності яких скоро закінчиться і вони продаються зі знижками. Це продукти, які повинні збуватися магазинами у першу чергу, аби не нести за собою матеріальних збитків для бізнесу та не продукувати нових харчових відходів.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Об'єктом автоматизації є процес пропозиції продуктових товарів різних супермаркетів для кінцевого користувача. Результат роботи досягається за декілька етапів:

- Отримання детальної інформації про смакові вподобання користувача;
- Складання індивідуальних правил;
- Запуск набору правил для кожного користувача та отримання набору індивідуальних продуктів.

Даних про автоматизацію цих процесів в Україні немає.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Система повинна виконувати наступні функції:

- Авторизація та автентифікація користувачів;
- Перегляд та редагування особистого профілю;
- Заповнення анкети продуктових обмежень;
- Генерація набору рекомендованих продуктів;
- Перегляд продуктів по категоріях;
- Перегляд усіх продуктів;
- Додавання продуктів до списку улюблених

4.2 Вимоги до надійності

Система повинна оброблювати всі виключні ситуації, що пов’язані з некоректними отриманими даними.

Усі дані у системі є конфіденційною інформацією і не можуть бути розголошеними.

4.3 Вимоги до складу і параметрів технічних засобів

Мінімальні вимоги до мобільного девайсу для коректної роботи системи:

- оперативна пам’ять – 100 МБ;
- версія Android – 7.0 або вище

Програмні засоби, що використовуються для проведення тестування:

- операційна система Windows 10, Android Studio Emulator;

Технічні засоби, що використовуються для проведення тестування:

- оперативна пам’ять – 8 Гб;
- жорсткий диск – 256 Гб;
- процесор – Intel Core i7 2,2 ГГц;
- графічна карта – Nvidia GeForce GTX 1050

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Таблиця 5.1 – Етапи розробки

№	Назва	Результат виконання
1.	Постановка задачі	Технічне завдання
2.	Розробка математичної моделі	Розроблена математична модель
3.	Генерація тестових даних	Розроблена інтеграція із серверами-джерелами даних
4.	Розробка алгоритму	Розроблений алгоритм
5.	Тестування алгоритму	Протестований алгоритм
6.	Розробка інтерфейсу	Розроблений інтерфейс
7.	Тестування інтерфейсу та алгоритму	Коректна робота інтерфейсу та алгоритму разом
8.	Оформлення ПЗ	Готова пояснювальна записка

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Перелік документів, що повинні бути представлені на контроль:

- технічне завдання;
- пояснювальна записка опису інформаційної системи для управління залишками продуктових магазинів;
- інформаційна система для управління залишками продуктових магазинів;
- тестові сценарії;
- керівництво користувача.

.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1003947965

Дата перевірки:
11.06.2020 01:42:53 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2020 02:11:13 EEST

ID користувача:
77149

Назва документу: Kozlova_is61

ID файлу: 1003963121 Кількість сторінок: 64 Кількість слів: 8654 Кількість символів: 64342 Розмір файлу: 1.24 MB

10.3% Схожість

Найбільша схожість: 3.11% з джерело бібліотеки. ID файлу: 12191135

4.03% Схожість з Інтернет джерелами	87	Page 66
10% Текстові збіги по Бібліотеці акаунту	295	Page 67

0% Цитат

Не знайдено жодних цитат

0% Вилучень

Вилучений текст відсутній

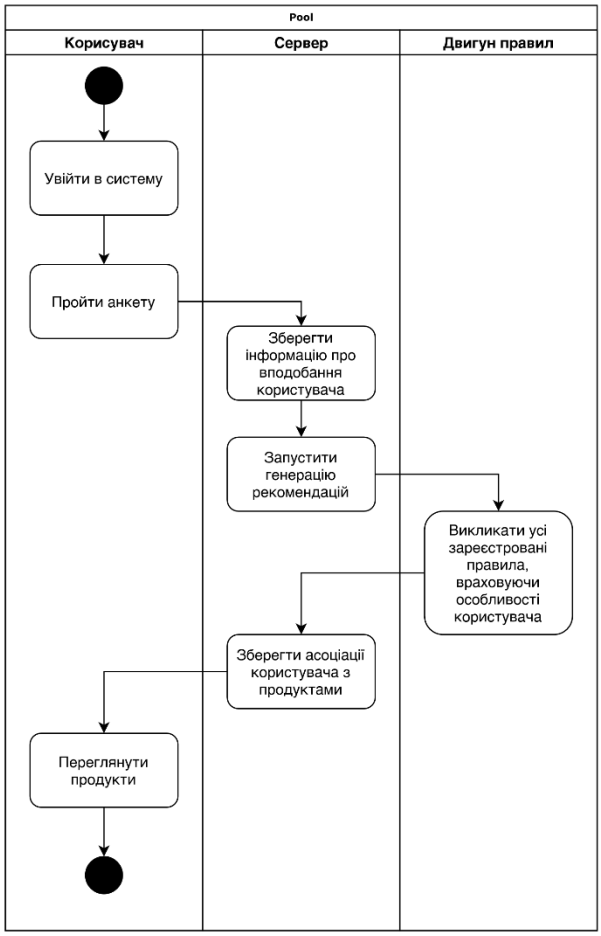
Підміна символів

Заміна символів	5

Графічний матеріал
до дипломного проєкту

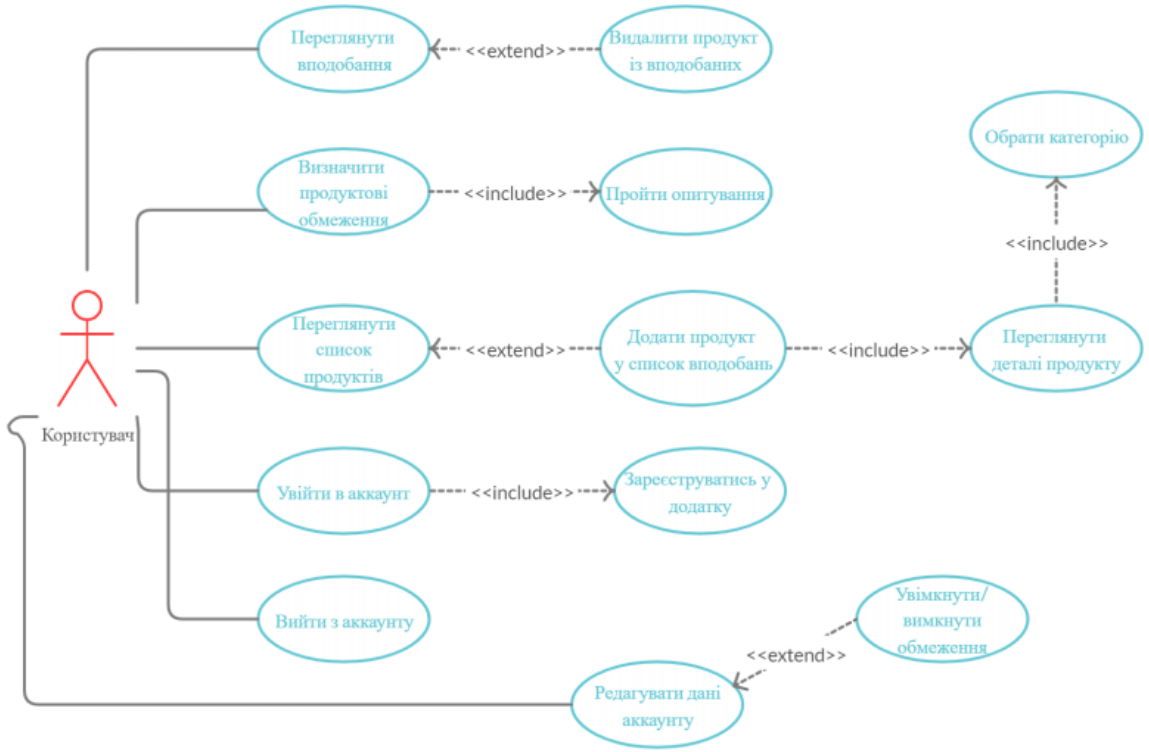
на тему: Інформаційна система для управління залишками продуктових магазинів

Київ – 2020 року



					ДП 6113.05.000 ССА	
Зм.	Арк.	№ докум.	Підп.	Дата	Схема структурна активності	Лит. / Маса / Масштаб
Розроб.		Козлова О.С.				
Перев.		Очеретяний О.К				Аркуш 1 / Аркушів 1
Т. Кон.						
Н. Кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41
Затв.		Очеретяний О.К				

ДП 6113.06.000 ССВ



					ДП 6113.06.000 ССВ		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використань	Літера	Маса / Масштаб
Розробив		Козлова О.С.					
Перевірив		Очеретяний О.К.				Аркуш 1	Аркушів 7
Т. кон.							
Н. кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Затвердив		Очеретяний О.К.					

ДП 6113.02.000 ССК

Powered by yFiles

					ДП 6113.02.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Козлова О.С.				
Перевірив		Очеретяний О.К				Аркуш 1 / Аркушів 7
Т. кон.						
Н. кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Очеретяний О.К				

ДП 6113.02.000 ССК

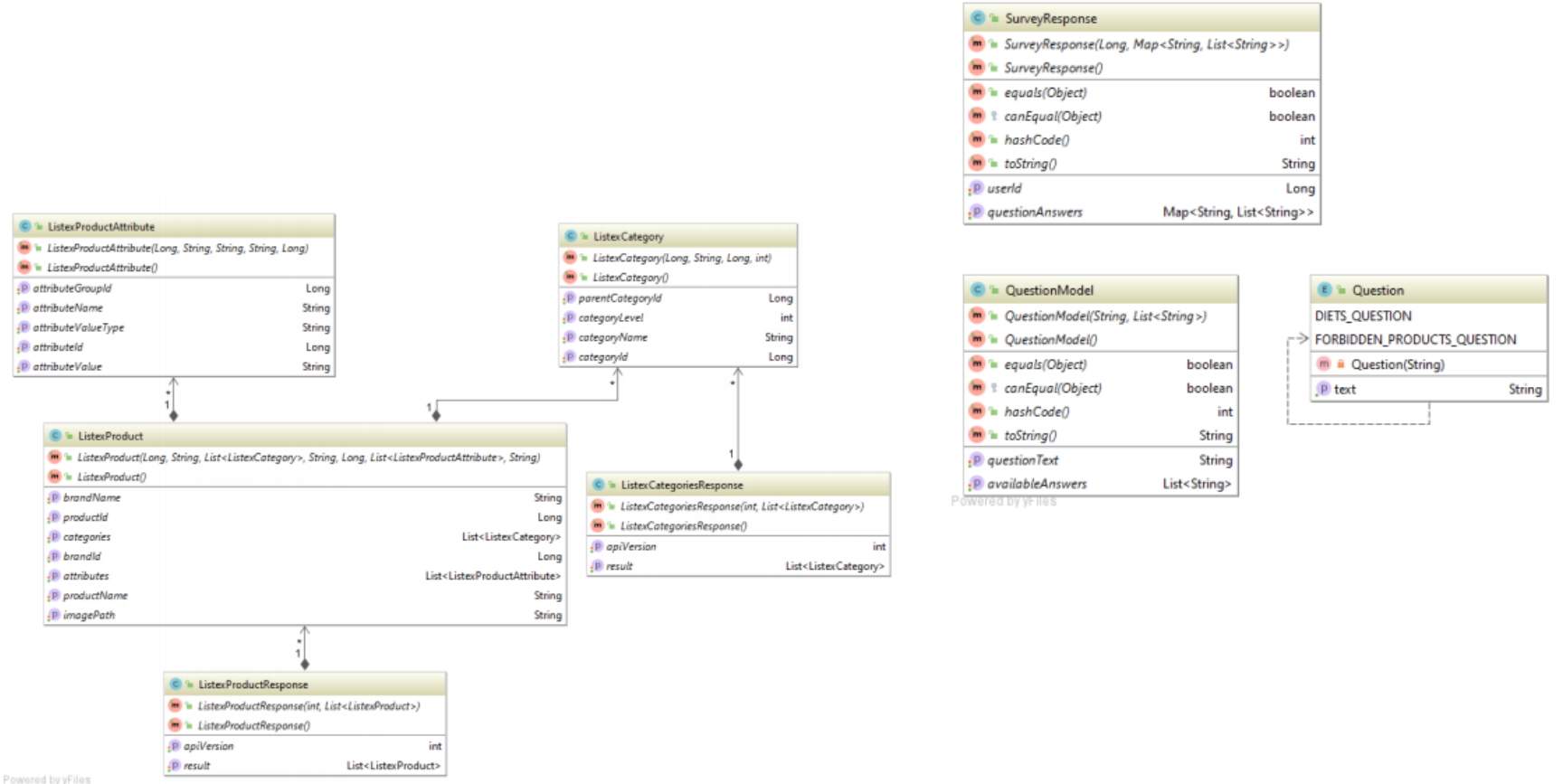
					ДП 6113.02.000 ССК
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення
Розробив		Козлова О.С.			Літера: — Маса: — Масштаб: —
Перевірив		Очеретяний О.К			Аркуш 2 Аркушів 7
Т. кон.					Інформаційна система для управління залишками продуктових магазинів
Н. кон.		Телишева Т.О.			КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Очеретяний О.К			

ДП 6113.02.000 ССК

					ДП 6113.02.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Козлова О.С.				
Перевірив		Очеретяний О.К.				Аркуш 3 / Аркушів 7
Т. кон.						

ДП 6113.02.000 ССК

					ДП 6113.02.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Козлова О.С.				
Перевірив		Очеретяний О.К				Аркуш 4 / Аркушів 7
Т. кон.						
Н. кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Очеретяний О.К				



					ДП 6113.02.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Козлова О.С.				
Перевірив		Очеретяний О.К.				Аркуш 5 / Аркушів 7
Т. кон.						
Н. кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Очеретяний О.К.				

ДП 6113.02.000 ССК

					ДП 6113.02.000 ССК		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера	Маса / Масштаб
Розробив		Козлова О.С.					
Перевірив		Очеретяний О.К				Аркуш 6	Аркушів 7
Т. кон.							
Н. кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Затвердив		Очеретяний О.К					

ДП 6113.02.000 ССК

					ДП 6113.02.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера: — / Маса: — / Масштаб: —
Розробив		Козлова О.С.				Аркуш 7 / Аркушів 7
Перевірив		Очеретяний О.К				
Т. кон.						
					Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Н. кон.		Тепишева Т.О.				
Затвердив		Очеретяний О.К				

					ДП 6113.04.000 ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Козлова О.С.				
Перевірив		Очеретяний О.К				Аркуш 130 / Аркушів 1
Т. кон.						
Н. кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Очеретяний О.К				

					ДП 6113.07.000 СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера / Маса / Масштаб
Розробив		Козлова О.С.				
Перевірив		Очеретяний О.К				Аркуш 1 / Аркушів 7
Т. кон.						
Н. кон.		Телишева Т.О.			Інформаційна система для управління залишками продуктових магазинів	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Затвердив		Очеретяний О.К				